# Multi-Layer Transformers Gradient Can be Approximated in Almost Linear Time

**Yingyu Liang** YINGYUL@HKU.HK; YLIANG@CS.WISC.EDU
**Zhizhou Sha** SHAZZ20@MAILS.TSINGHUA.EDU.CN
**Zhenmei Shi** ZHMEISHI@CS.WISC.EDU
**Zhao Song** MAGIC.LINUXKDE@GMAIL.COM
**Yufa Zhou** YUFAZHOU@SEAS.UPENN.EDU

## Abstract

The quadratic computational complexity in the self-attention mechanism of popular transformer architectures poses significant challenges for training and inference, particularly in terms of efficiency and memory requirements. Towards addressing these challenges, this paper introduces a novel fast computation method for gradient calculation in multi-layer transformer models. Our approach enables the computation of gradients for the entire multi-layer transformer model in almost linear time $n^{1+o(1)}$, where $n$ is the input sequence length. This breakthrough significantly reduces the computational bottleneck associated with the traditional quadratic time complexity. Our theory holds for any loss function and maintains a bounded approximation error across the entire model. Furthermore, our analysis can hold when the multi-layer transformer model contains many practical sub-modules, such as residual connection, casual mask, and multi-head attention. By improving the efficiency of gradient computation in large language models, we hope that our work will facilitate the more effective training and deployment of long-context language models based on our theoretical results.

## 1. Introduction

Large Language Models (LLMs), such as ChatGPT [99], GPT-4 [1], Claude 3.5 [7], Llama 3.1 [83, 112], and others, have demonstrated immense potential to enhance various aspects of our daily lives, e.g., conversation AI [78], AI agent [18, 124], search AI [93], AI assistant [29, 57, 89, 135] and many so on. One of the most emergent abilities of LLMs is dealing with long-context information, a format that is crucial for recording material like academic papers, official reports, legal documents, and so on. LLMs have proven adept at tackling long-context tasks, including Retrieval Augmented Generation (RAG) [37, 61], zero-shot summarization [19, 81, 139, 142], and maintaining very long-term conversations [88, 128, 130], and so on. This proficiency has necessitated the development of long-context modeling capabilities within LLMs.

LLMs are mainly based on Transformer architecture, the key module of which is the self-attention mechanism. In attention computation, we will compute the attention score between each pair of tokens, which is the complexity bottleneck during long context training and inference. In detail, we need to spend $O(n^2d)$ running time for each self-attention block, which is quadratic in $n$, where $n$ is the length of the context token and $d$ is the hidden feature dimension of the model. For example, LLaMA 3.1 405B [83], one of the cutting-edge LLMs, supports $n =$128k and $d = 4096$ taking 30.84M GPU training hours, which underscores the need for more efficient training processes for such extensive context models. Given the extensive context lengths of LLMs, this quadratic scaling results in several critical challenges: (1) a marked decrease in training efficiency [43, 51, 86]; (2) substantial memory requirements to accommodate the large quadratic gradient matrices [63, 85]; and (3) significant energy usage, which in turn contributes to higher carbon dioxide emissions [98, 106].

One seminal work [3] showed that the self-attention inference can be approximated in almost linear time. However, this result is for the *inference* time but does not address the main challenge, which is the expensive computation in the *training* time. In this work, we address this main challenge, by proving that the quadratic computational complexity in the back-propagation of self-attention can be approximated in almost linear time. This suggests we may be able to save the substantial resources required for training Large Language Models.

### 1.1. Key background

We first introduce a basic background, starting with defining the softmax function.

**Definition 1 (Softmax)** *Let $z \in \mathbb{R}^n$. We define $\mathsf{Softmax} : \mathbb{R}^n \to \mathbb{R}^n$ satisfying*

$$\mathsf{Softmax}(z) := \exp(z)/\langle \exp(z), \mathbf{1}_n \rangle.$$

*Here we apply $\exp$ to a vector entry-wise.*

Then, we introduce the definition of a single self-attention layer.

**Definition 2 (Single layer self-attention)** *Let $n$ be the number of tokens and $d$ be the hidden dimension size. Let $X \in \mathbb{R}^{n \times d}$ denote the input data matrix. Let $Q := XW_Q, K := XW_K, V := XW_V \in \mathbb{R}^{n \times d}$. Let $W := W_Q W_K^\top \in \mathbb{R}^{d \times d}$ be the key-query weight matrix, and let $W_V \in \mathbb{R}^{d \times d}$ be the value weight matrix. The self-attention function $\mathsf{Attn}(Q, K, V)$ is:*

$$\mathsf{Attn}(Q, K, V) = \mathsf{Softmax}(QK^\top/d)V.$$

*We remark that we apply $\mathsf{Softmax}$ to each row of the $n \times n$ matrix.*

*The above $\mathsf{Attn}$ function takes $Q, K, V$ as parameters, and can be re-parameterized by input matrix $X$ and then re-written in the following sense*

$$\mathsf{Attn}(X) := f(X) \cdot X \cdot W_V,$$

*where (1) $A := \exp(XWX^\top/d) \in \mathbb{R}^{n \times n}$ and $\exp$ is applied element-wise, (2) $D := \mathrm{diag}(A\mathbf{1}_n) \in \mathbb{R}^{n \times n}$, and (3) $f(X) := D^{-1}A \in \mathbb{R}^{n \times n}$ is the attention matrix.*

In contemporary LLMs, the architecture typically incorporates multiple layers of transformers. Consequently, in order to design a fast training algorithm for the entire model, it is imperative to examine the self-attention layers within these multi-layer transformer structures. We provide a formal definition for multi-layer self-attention as follows.

**Definition 3 (Multi-layer transformer)** *Let $m$ denote the number of transformer layers in the model. Let $X$ be the input sentence. Let $g_i$ denote components other than self-attention in the $i$-th* *transformer layer. Let* $\mathsf{Attn}_i$ *denote the self-attention module in the* $i$*-th transformer layer (see also Definition 2). We define an* $m$*-layer transformer as*

$$\mathsf{F}_m(X) := g_m \circ \mathsf{Attn}_m \circ g_{m-1} \circ \mathsf{Attn}_{m-1} \circ \cdots \circ g_1 \circ \mathsf{Attn}_1 \circ g_0(X),$$

*where* $\circ$ *denotes function composition.*

In Definition 3, the $g_i$ includes the layer norm, MLP, residual connection, dropout, positional encoding, multi-head concatenation, and other operations. All forward and backward computations of these modules can be run in linear time with respect to $n$. Thus, in this work, we mainly focus on the acceleration of self-attention module. Specifically, as shown in Definition 2, the $n \times n$ attention matrix $f(X)$ dominates the computational complexity, introducing a quadratic bottleneck. In the exact computation case, if the attention matrix is full rank, no acceleration is possible. However, by compromising negligible accuracy, designing a fast sub-quadratic algorithm becomes feasible. Fortunately, by employing the polynomial kernel approximation method from Aggarwal and Alman [2], we can approximate the attention matrix and achieve an almost linear time $n^{1+o(1)}$ algorithm, effectively breaking the quadratic bottleneck.

### 1.2. Our contributions

We now state our main contributions as follows:

**Theorem 4 (Main result, informal version of Theorem 9)** *Let* $n$ *be the number of tokens, and* $d$ *be the hidden dimension size. We assume* $d = O(\log n)$ *and each number in matrices can be written using* $O(\log n)$ *bits. There exists an algorithm (Algorithm 1) that can compute the gradient of multi-layer self-attention (see also Definition 3) in almost linear time* $n^{1+o(1)}d = n^{1+o(1)}$*, where the approximation error of entire model can be bounded in* $1/\operatorname{poly}(n)$*.*

Our assumption is mild when the context length $n$ is large, as we usually see feature dimension $d$ as a constant, which is also used in Alman and Song [3, 4]. Our results indicate that large language models (LLMs) can be trained in almost linear time $n^{1+o(1)}$ and maintain a robust approximation guarantee, while the traditional way takes $\Omega(n^2)$ time. This advancement is realized through the application of polynomial kernel approximation [3, 4]. To be more specific, by leveraging the inherent sparsity within the dense attention matrix, we are able to perform efficient low-rank approximation, thereby significantly accelerating the computation of the dense matrices. Our framework is applicable to *any* loss function, making it universally applicable. Furthermore, our analysis can hold when the multi-layer transformer model contains many practical sub-modules, such as residual connection, casual mask, and multi-head attention (Section 4).

Our contributions are as follows:

- We introduce a fast computation method that allows the gradient of each self-attention layer to be approximated in almost linear time $n^{1+o(1)}$ with $1/\operatorname{poly}(n)$ error, breaking the quadratic time complexity bottleneck (Theorem 8).
- We extend our single-layer results to module-wise gradient computation so that our Algorithm 1 can provide approximated gradient computation in $m \cdot n^{1+o(1)}$ time for $m$-layer transformer. Furthermore, the approximation of the gradient diverges from the exact gradient by an error of $1/\operatorname{poly}(n)$ when considered across the entire model (Theorem 9).

- Additionally, our analysis can hold when the multi-layer transformer model contains residual connection, casual mask, and multi-head attention (Section 4).

## 2. Preliminary

**Notations** For any positive integer $n$, we use $[n]$ to denote set $\{1, 2, \cdots, n\}$. For two vectors $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^n$, we use $\langle x, y \rangle$ to denote the inner product between $x, y$. Namely, $\langle x, y \rangle = \sum_{i=1}^n x_i y_i$. We use $e_i$ to denote a vector where only $i$-th coordinate is 1, and other entries are 0. For each $a, b \in \mathbb{R}^n$, we use $a \odot b \in \mathbb{R}^n$ to denote the Hardamard product, i.e. the $i$-th entry of $(a \odot b)$ is $a_i b_i$ for all $i \in [n]$. We use $\mathbf{1}_n$ to denote a length-$n$ vector where all the entries are ones. We use $\|A\|_\infty$ to denote the $\ell_\infty$ norm of a matrix $A \in \mathbb{R}^{n \times d}$, i.e. $\|A\|_\infty := \max_{i \in [n], j \in [d]} |A_{i,j}|$. We use $\mathrm{poly}(n)$ to denote polynomial time complexity with respective to $n$.

### 2.1. Loss function

The loss function is the optimization objective in the training of LLMs, and we define it as follows.

**Definition 5 (Loss function $L(X)$)** *For some input matrix $X \in \mathbb{R}^{n \times d}$, we define the one-unit loss function $\ell(X)_{j,k} : \mathbb{R}^{n \times d} \to \mathbb{R}$, for any $j \in [n], k \in [d]$. Furthermore, we define the overall loss function $L(X)$, such that*

$$L(X) = \sum_{j=1}^{n} \sum_{k=1}^{d} \ell(X)_{j,k}$$

### 2.2. Close forms of gradient components

In training large language models (LLMs), updating the model necessitates computing the gradient of weights for every layer. Consequently, it becomes essential to derive the closed-form expressions for all corresponding gradient components with respect to the weights of the query, key, and value matrices in the transformer model. We first define some intermediate variables before detailing these gradient components in each self-attention transformer layer.

**Definition 6 (Intermediate variables $T_i$)** *Let $m$ denote the number of transformer layers in the model. Let $m$-layer self-attention transformer be defined as Definition 3. Let $d$ denote the hidden dimension. Let $n$ denote the sequence length. Let $X \in \mathbb{R}^{n \times d}$ be the input sentence. Let $g_i$ denote components other than self-attention in the $i$-th transformer layer. Let $\mathsf{Attn}_i$ denote the self-attention module in the $i$-th transformer layer (see also Definition 2).*

*For $i \in \{0, 1, 2, \cdots, m\}$, we define $T_i(X) \in \mathbb{R}^{n \times d}$ be the intermediate variable (hidden states) output by $i$-th layer self-attention transformer. Namely, we have*

$$T_i(X) = \begin{cases} g_0(X), & i = 0; \\ (g_i \circ \mathsf{Attn}_i)(T_{i-1}(X)), & i \in [m]. \end{cases}$$

*Here, we use $\circ$ to denote function composition.*

Then, we are ready to introduce the close forms of the three gradient components in a single self-attention transformer layer. Notably, according to the chain rule, the gradient of the $k$-th transformer layer in LLMs depends on the gradient components from the $(k+1)$-th transformer layer. The gradient can be calculated for every transformer layer by combining the upstream and local gradients. The close forms of the gradients for each layer in multi-layer transformers are formalized in the following lemma (Lemma 7).

**Lemma 7 (Close form of gradient components, informal version of Lemma 18)** *Let $L(X)$ be defined as Definition 5. Let $W_i = W_{Q_i} W_{K_i}^\top, W_{V_i} \in \mathbb{R}^{d \times d}$ denote the attention weight in the $i$-th transformer layer. Let $T_i(X)$ denote the intermediate variable output by $i$-th self-attention transformer layer (see Definition 6). Let $G_i \in \mathbb{R}^{n \times d}$ denote the gradient matrix resulting from the application of the chain rule up to the function $g_i$, i.e., $G_i = \frac{\mathrm{d}L(X)}{\mathrm{d}\mathsf{Attn}_i(T_{i-1}(X))}$. For $j \in [n], k \in [d]$, let $G_i(j,k)$ denote the $(j,k)$-th entry of $G_i$, let $\frac{\mathrm{d}\mathsf{Attn}_i(T_{i-1}(X))_{j,k}}{\mathrm{d}T_{i-1}(X)} \in \mathbb{R}^{n \times d}$ denote the gradient of $(j,k)$-th entry of $\mathsf{Attn}_i(T_{i-1}(X))$. Then, we can show that*

- ***Part 1.***

$$\frac{\mathrm{d}L(X)}{\mathrm{d}T_{i-1}(X)} = \sum_{j=1}^{n} \sum_{k=1}^{d} G_i(j,k) \cdot \frac{\mathrm{d}\mathsf{Attn}_i(T_{i-1}(X))_{j,k}}{\mathrm{d}T_{i-1}(X)}.$$

- ***Part 2.***

$$\frac{\mathrm{d}L(X)}{\mathrm{d}W_i} = \sum_{j=1}^{n} \sum_{k=1}^{d} G_i(j,k) \cdot \frac{\mathrm{d}\mathsf{Attn}_i(T_{i-1}(X))_{j,k}}{\mathrm{d}W_i}.$$

- ***Part 3.***

$$\frac{\mathrm{d}L(X)}{\mathrm{d}W_{V_i}} = \sum_{j=1}^{n} \sum_{k=1}^{d} G_i(j,k) \cdot \frac{\mathrm{d}\mathsf{Attn}_i(T_{i-1}(X))_{j,k}}{\mathrm{d}W_{V_i}}.$$

Based on the above close forms of three gradient components, we can move on to our main results smoothly.

## 3. Main Result

### 3.1. Fast computing for single layer

In the case of single-layer attention, we provide our theorem that state the three gradient components can be calculated in almost linear time with negligible error.

**Theorem 8 (Single-layer gradient approximation)** *We assume $d = O(\log n)$ and each number in matrices can be written using $O(\log n)$ bits. Let $L(X)$ be defined as Definition 5. Suppose we have a single-layer self-attention transformer model ($m = 1$ in Definition 3). We can approximate one-layer self-attention for three gradient components, i.e. $\frac{\mathrm{d}L(X)}{\mathrm{d}X}$, $\frac{\mathrm{d}L(X)}{\mathrm{d}W}$ and $\frac{\mathrm{d}L(X)}{\mathrm{d}W_V}$, in $n^{1+o(1)}$ time with $1/\operatorname{poly}(n)$ error.*

**Proof** We finish the proof by Lemma 11, 12 and 13. ■

### 3.2. Fast computing for multi-layer transformers

Based on the results demonstrated in previous sections, we are ready to introduce our main result: the whole transformer model can be approximated in almost linear time.

**Theorem 9 (Main result, formal version of Theorem 4)** *Let $m$ denote the number of transformer layers. We assume $d = O(\log n)$ and each number in matrices can be written using $O(\log n)$ bits. We can show that, for any $i \in [m]$, all the gradient components (see also Lemma 7) of the $i$-th layer can be computed by Algorithm 1 in almost linear time $n^{1+o(1)}$, and the approximation error of entire $m$ layer transformer model can be bounded by $1/\operatorname{poly}(n)$.*

**Proof** The theorem can be proved by directly combining Theorem 8 and Lemma 15. ■

Theorem 9 demonstrates that, during the training of a multi-layer transformer model, at each training iteration, the gradient computation for the weight matrices of each layer can be performed in almost linear time $n^{1+o(1)}$. This result supports the feasibility of fast training for any transformer-based LLMs. We achieve acceleration in multi-layer transformer networks primarily through the application of the chain rule during back-propagation. By extending our single-layer transformer result to each individual layer, we ensure that gradients can be efficiently propagated from the final loss $L(X)$ to the initial input $X$. This process is accomplished within almost linear time. For a detailed illustration, please refer to Algorithm 1.

## 4. Extension

**Multi-head attention and residual connections.** Multi-head attention and residual connections are important components in attention mechanisms. While these components were not involved in our initial analysis for simplicity, incorporating them into our algorithm is straightforward. The detailed analysis of incorporating residual connection with our framework can be found in Section M and Lemma 85. For the synergy with multi-head attention, we provide comprehensive analysis in Section N and Lemma 89. Our algorithm maintains the capability to compute gradients for multi-layer transformers with multi-head attention and residual connection in almost linear time, suggesting that it can be readily adapted to more practical transformer models.

**Causal attention mask.** The causal attention mask is critical to prevent transformers from "cheating" during training by ensuring future information is not used. The full-rank characteristic of the causal attention mask poses challenges for low-rank approximations. Nevertheless, we have identified a method to accelerate the computation of causal masked attention by exploiting its inherent properties, as demonstrated in Liang et al. [72], remaining almost linear time complexity. More detailed analysis can be found in Section L and Lemma 81 and 82.

## 5. Conclusion

The attention mechanism in transformer models inherently has quadratic time complexity with respect to the input token length. In this work, we propose a novel Algorithm 1, which can approximately train a multi-layer transformer model in almost linear time, introducing only a small error. Moreover, our algorithm is compatible with any loss function, practical sub-modules (residual connection, casual mask, multi-head attention). We believe our theoretical findings will play an important role in accelerating the training of LLMs in the future.

## References


[1] 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.

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

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

[4] Josh Alman and Zhao Song. The fine-grained complexity of gradient computation for training large language models. *arXiv preprint arXiv:2402.04497*, 2024.

[5] 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*, 2024.

[6] Josh Alman, Timothy Chu, Aaron Schild, and Zhao Song. Algorithms and hardness for linear algebra on geometric graphs. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 541–552. IEEE, 2020.

[7] Anthropic. The claude 3 model family: Opus, sonnet, haiku. https://www-cdn.anthropic.com, 2024.

[8] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[9] Brian Bartoldson, Ari Morcos, Adrian Barbu, and Gordon Erlebacher. The generalization-stability tradeoff in neural network pruning. *Advances in Neural Information Processing Systems*, 33:20852–20864, 2020.

[10] Rouzbeh Behnia, Mohammadreza Reza Ebrahimi, Jason Pacheco, and Balaji Padmanabhan. Ew-tune: A framework for privately fine-tuning large language models with differential privacy. In *2022 IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 560–566. IEEE, 2022.

[11] Davis Blalock, Jose Javier Gonzalez Ortiz, Jonathan Frankle, and John Guttag. What is the state of neural network pruning? *Proceedings of machine learning and systems*, 2:129–146, 2020.

[12] Jan van den Brand, Zhao Song, and Tianyi Zhou. Algorithm and hardness for dynamic attention maintenance in large language models. *arXiv preprint arXiv:2304.02207*, 2023.

[13] Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D Lee, Deming Chen, and Tri Dao. Medusa: Simple llm inference acceleration framework with multiple decoding heads. *arXiv preprint arXiv:2401.10774*, 2024.

[14] Shang Chai, Liansheng Zhuang, and Fengying Yan. Layoutdm: Transformer-based diffusion model for layout generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18349–18358, 2023.

[15] Beidi Chen, Zichang Liu, Binghui Peng, Zhaozhuo Xu, Jonathan Lingjie Li, Tri Dao, Zhao Song, Anshumali Shrivastava, and Christopher Re. Mongoose: A learnable lsh framework for efficient neural network training. In *International Conference on Learning Representations*, 2020.

[16] Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. Extending context window of large language models via positional interpolation. *arXiv preprint arXiv:2306.15595*, 2023.

[17] Tianyi Chen, Bo Ji, Tianyu Ding, Biyi Fang, Guanyi Wang, Zhihui Zhu, Luming Liang, Yixin Shi, Sheng Yi, and Xiao Tu. Only train once: A one-shot neural network training and pruning framework. *Advances in Neural Information Processing Systems*, 34:19637–19651, 2021.

[18] Weize Chen, Ziming You, Ran Li, Yitong Guan, Chen Qian, Chenyang Zhao, Cheng Yang, Ruobing Xie, Zhiyuan Liu, and Maosong Sun. Internet of agents: Weaving a web of heterogeneous agents for collaborative intelligence. *arXiv preprint arXiv:2407.07061*, 2024.

[19] Anshuman Chhabra, Hadi Askari, and Prasant Mohapatra. Revisiting zero-shot abstractive summarization in the era of large language models from the perspective of position bias. *arXiv preprint arXiv:2401.01989*, 2024.

[20] Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*, 2023.

[21] Tri Dao and Albert Gu. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality. *arXiv preprint arXiv:2405.21060*, 2024.

[22] Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359, 2022.

[23] Yichuan Deng, Sridhar Mahadevan, and Zhao Song. Randomized and deterministic attention sparsification algorithms for over-parameterized feature dimension. *arXiv preprint arXiv:2304.04397*, 2023.

[24] Yichuan Deng, Zhao Song, Shenghao Xie, and Chiwun Yang. Unmasking transformers: A theoretical approach to data recovery via attention weights. *arXiv preprint arXiv:2310.12462*, 2023.

[25] Harry Dong, Xinyu Yang, Zhenyu Zhang, Zhangyang Wang, Yuejie Chi, and Beidi Chen. Get more with less: Synthesizing recurrence with kv cache compression for efficient llm inference. *arXiv preprint arXiv:2402.09398*, 2024.

[26] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly,

et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[27] Mostafa Elhoushi, Akshat Shrivastava, Diana Liskovich, Basil Hosmer, Bram Wasti, Liangzhen Lai, Anas Mahmoud, Bilge Acun, Saurabh Agarwal, Ahmed Roman, et al. Layer skip: Enabling early exit inference and self-speculative decoding. *arXiv preprint arXiv:2404.16710*, 2024.

[28] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first International Conference on Machine Learning*, 2024.

[29] Tao Feng, Chuanyang Jin, Jingyu Liu, Kunlun Zhu, Haoqin Tu, Zirui Cheng, Guanyu Lin, and Jiaxuan You. How far are we from agi. *arXiv preprint arXiv:2405.10313*, 2024.

[30] Quentin Fournier, Gaétan Marceau Caron, and Daniel Aloise. A practical survey on faster and lighter transformers. *ACM Computing Surveys*, 55(14s):1–40, 2023.

[31] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

[32] Elias Frantar and Dan Alistarh. Optimal brain compression: A framework for accurate post-training quantization and pruning. *Advances in Neural Information Processing Systems*, 35: 4475–4488, 2022.

[33] Elias Frantar and Dan Alistarh. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *International Conference on Machine Learning*, pages 10323–10337. PMLR, 2023.

[34] Yeqi Gao, Sridhar Mahadevan, and Zhao Song. An over-parameterized exponential regression. *arXiv preprint arXiv:2303.16504*, 2023.

[35] Yeqi Gao, Zhao Song, Xin Yang, and Ruizhe Zhang. Fast quantum algorithm for attention computation. *arXiv preprint arXiv:2307.08045*, 2023.

[36] Yeqi Gao, Zhao Song, Xin Yang, and Yufa Zhou. Differentially private attention computation. In *Neurips Safe Generative AI Workshop 2024*, 2024.

[37] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2023.

[38] Suyu Ge, Yunan Zhang, Liyuan Liu, Minjia Zhang, Jiawei Han, and Jianfeng Gao. Model tells you what to discard: Adaptive kv cache compression for llms. *arXiv preprint arXiv:2310.01801*, 2023.

[39] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

[40] Kamal Gupta, Justin Lazarow, Alessandro Achille, Larry S Davis, Vijay Mahadevan, and Abhinav Shrivastava. Layouttransformer: Layout generation and completion with self-attention. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1004–1014, 2021.

[41] Insu Han, Rajesh Jayaram, Amin Karbasi, Vahab Mirrokni, David Woodruff, and Amir Zandieh. Hyperattention: Long-context attention in near-linear time. In *The Twelfth International Conference on Learning Representations*, 2024.

[42] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28, 2015.

[43] Nan He, Hanyu Lai, Chenyang Zhao, Zirui Cheng, Junting Pan, Ruoyu Qin, Ruofan Lu, Rui Lu, Yunchen Zhang, Gangming Zhao, et al. Teacherlm: Teaching to fish rather than giving the fish, language modeling likewise. *arXiv preprint arXiv:2310.19019*, 2023.

[44] Torsten Hoefler, Dan Alistarh, Tal Ben-Nun, Nikoli Dryden, and Alexandra Peste. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *Journal of Machine Learning Research*, 22(241):1–124, 2021.

[45] Jerry Yao-Chieh Hu, Donglin Yang, Dennis Wu, Chenwei Xu, Bo-Yu Chen, and Han Liu. On sparse modern hopfield model. In *Thirty-seventh Conference on Neural Information Processing Systems (NeurIPS)*, 2023.

[46] Jerry Yao-Chieh Hu, Pei-Hsuan Chang, Haozheng Luo, Hong-Yu Chen, Weijian Li, Wei-Po Wang, and Han Liu. Outlier-efficient hopfield layers for large transformer-based models. In *Forty-first International Conference on Machine Learning (ICML)*, 2024.

[47] Jerry Yao-Chieh Hu, Bo-Yu Chen, Dennis Wu, Feng Ruan, and Han Liu. Nonparametric modern hopfield models. *arXiv preprint arXiv:2404.03900*, 2024.

[48] Jerry Yao-Chieh Hu, Thomas Lin, Zhao Song, and Han Liu. On computational limits of modern hopfield models: A fine-grained complexity analysis. In *Forty-first International Conference on Machine Learning (ICML)*, 2024.

[49] Jerry Yao-Chieh Hu, Maojiang Su, En-Jui Kuo, Zhao Song, and Han Liu. Computational limits of low-rank adaptation (lora) for transformer-based models. *arXiv preprint arXiv:2406.03136*, 2024.

[50] 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.

[51] Shengding Hu, Yuge Tu, Xu Han, Chaoqun He, Ganqu Cui, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, Weilin Zhao, et al. Minicpm: Unveiling the potential of small language models with scalable training strategies. *arXiv preprint arXiv:2404.06395*, 2024.

[52] Itay Hubara, Brian Chmiel, Moshe Island, Ron Banner, Joseph Naor, and Daniel Soudry. Accelerated sparse neural training: A provable and efficient method to find n: m transposable masks. *Advances in neural information processing systems*, 34:21099–21111, 2021.

[53] Tian Jin, Michael Carbin, Dan Roy, Jonathan Frankle, and Gintare Karolina Dziugaite. Pruning's effect on generalization through the lens of training and regularization. *Advances in Neural Information Processing Systems*, 35:37947–37961, 2022.

[54] Praneeth Kacham, Vahab Mirrokni, and Peilin Zhong. Polysketchformer: Fast transformers via sketches for polynomial kernels. *arXiv preprint arXiv:2310.01655*, 2023.

[55] Feyza Duman Keles, Pruthuvi Mahesakya Wijewardena, and Chinmay Hegde. On the computational complexity of self-attention. In *International Conference on Algorithmic Learning Theory*, pages 597–619. PMLR, 2023.

[56] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020.

[57] Tzu-Sheng Kuo, Aaron Lee Halfaker, Zirui Cheng, Jiwoo Kim, Meng-Hsin Wu, Tongshuang Wu, Kenneth Holstein, and Haiyi Zhu. Wikibench: Community-driven data curation for ai evaluation on wikipedia. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, pages 1–24, 2024.

[58] N Lee, T Ajanthan, and P Torr. Snip: single-shot network pruning based on connection sensitivity. In *International Conference on Learning Representations*. Open Review, 2019.

[59] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, 2021.

[60] Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pages 19274–19286. PMLR, 2023.

[61] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474, 2020.

[62] Chenyang Li, Yingyu Liang, Zhenmei Shi, Zhao Song, and Tianyi Zhou. Fourier circuits in neural networks: Unlocking the potential of large language models in mathematical reasoning and modular arithmetic. *arXiv preprint arXiv:2402.09469*, 2024.

[63] Dacheng Li, Hongyi Wang, Eric Xing, and Hao Zhang. Amp: Automatically finding model parallel strategies with heterogeneity awareness. *Advances in Neural Information Processing Systems*, 35:6630–6639, 2022.

[64] Minghan Li, Xilun Chen, Ari Holtzman, Beidi Chen, Jimmy Lin, Wen-tau Yih, and Xi Victoria Lin. Nearest neighbor speculative decoding for llm generation and attribution. *arXiv preprint arXiv:2405.19325*, 2024.

[65] Tianle Li, Ge Zhang, Quy Duc Do, Xiang Yue, and Wenhu Chen. Long-context llms struggle with long in-context learning. *arXiv preprint arXiv:2404.02060*, 2024.

[66] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, 2021.

[67] Xiaoyu Li, Yingyu Liang, Zhenmei Shi, and Zhao Song. A tighter complexity analysis of sparsegpt. *arXiv preprint arXiv:2408.12151*, 2024.

[68] Yanghao Li, Hanzi Mao, Ross Girshick, and Kaiming He. Exploring plain vision transformer backbones for object detection. In *European conference on computer vision*, pages 280–296. Springer, 2022.

[69] Yuchen Li, Yuanzhi Li, and Andrej Risteski. How do transformers learn topic structure: Towards a mechanistic understanding. In *International Conference on Machine Learning*, pages 19689–19729. PMLR, 2023.

[70] Yucheng Li, Bo Dong, Chenghua Lin, and Frank Guerin. Compressing context to enhance inference efficiency of large language models. *arXiv preprint arXiv:2310.06201*, 2023.

[71] Zhihang Li, Zhao Song, and Tianyi Zhou. Solving regularized exp, cosh and sinh regression problems. *arXiv preprint arXiv:2303.15725*, 2023.

[72] 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*, 2024.

[73] Yingyu Liang, Zhizhou Sha, Zhenmei Shi, and Zhao Song. Differential privacy mechanisms in neural tangent kernel regression. *arXiv preprint arXiv:2407.13621*, 2024.

[74] Yingyu Liang, Zhenmei Shi, Zhao Song, and Chiwun Yang. Toward infinite-long prefix in transformer. *arXiv preprint arXiv:2406.14036*, 2024.

[75] Yingyu Liang, Zhenmei Shi, Zhao Song, and Yufa Zhou. Unraveling the smoothness properties of diffusion models: A gaussian mixture perspective. *arXiv preprint arXiv:2405.16418*, 2024.

[76] Yingyu Liang, Zhenmei Shi, Zhao Song, and Yufa Zhou. Differential privacy of cross-attention with provable guarantee. *arXiv preprint arXiv:2407.14717*, 2024.

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

[78] Na Liu, Liangyu Chen, Xiaoyu Tian, Wei Zou, Kaijiang Chen, and Ming Cui. From llm to conversational agent: A memory enhanced architecture with fine-tuning of large language models. *arXiv preprint arXiv:2401.02777*, 2024.

[79] Tianlin Liu and Friedemann Zenke. Finding trainable sparse networks through neural tangent transfer. In *International Conference on Machine Learning*, pages 6336–6347. PMLR, 2020.

[80] Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, 2022.

[81] Yixin Liu, Kejian Shi, Katherine S He, Longtian Ye, Alexander R Fabbri, Pengfei Liu, Dragomir Radev, and Arman Cohan. On learning to summarize with large language models as references. *arXiv preprint arXiv:2305.14239*, 2023.

[82] Zichang Liu, Jue Wang, Tri Dao, Tianyi Zhou, Binhang Yuan, Zhao Song, Anshumali Shrivastava, Ce Zhang, Yuandong Tian, Christopher Re, et al. Deja vu: Contextual sparsity for efficient llms at inference time. In *International Conference on Machine Learning*, pages 22137–22176. PMLR, 2023.

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

[84] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

[85] Kai Lv, Hang Yan, Qipeng Guo, Haijun Lv, and Xipeng Qiu. Adalomo: Low-memory optimization with adaptive learning rate. *arXiv preprint arXiv:2310.10195*, 2023.

[86] Kai Lv, Yuqing Yang, Tengxiao Liu, Qinghui Gao, Qipeng Guo, and Xipeng Qiu. Full parameter fine-tuning for large language models with limited resources. *arXiv preprint arXiv:2306.09782*, 2023.

[87] Nanye Ma, Mark Goldstein, Michael S Albergo, Nicholas M Boffi, Eric Vanden-Eijnden, and Saining Xie. Sit: Exploring flow and diffusion-based generative models with scalable interpolant transformers. *arXiv preprint arXiv:2401.08740*, 2024.

[88] Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating very long-term conversational memory of llm agents. *arXiv preprint arXiv:2402.17753*, 2024.

[89] Amama Mahmood, Junxiang Wang, Bingsheng Yao, Dakuo Wang, and Chien-Ming Huang. Llm-powered conversational voice assistants: Interaction patterns, opportunities, challenges, and design guidelines. *arXiv preprint arXiv:2309.13879*, 2023.

[90] Jean Mercat, Igor Vasiljevic, Sedrick Keh, Kushal Arora, Achal Dave, Adrien Gaidon, and Thomas Kollar. Linearizing large language models. *arXiv preprint arXiv:2405.06640*, 2024.

[91] Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. Rethinking the role of demonstrations: What makes in-context learning work? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11048–11064, 2022.

[92] Jesse Mu, Xiang Li, and Noah Goodman. Learning to compress prompts with gist tokens. *Advances in Neural Information Processing Systems*, 36, 2024.

[93] OpenAI. Searchgpt prototype. https://openai.com/index/searchgpt-prototype, 2024.

[94] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4195–4205, 2023.

[95] Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, Huanqi Cao, Xin Cheng, Michael Chung, Matteo Grella, et al. Rwkv: Reinventing rnns for the transformer era. *arXiv preprint arXiv:2305.13048*, 2023.

[96] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI blog*, 2019.

[97] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[98] Siddharth Samsi, Dan Zhao, Joseph McDonald, Baolin Li, Adam Michaleas, Michael Jones, William Bergeron, Jeremy Kepner, Devesh Tiwari, and Vijay Gadepally. From words to watts: Benchmarking the energy costs of large language model inference. In *2023 IEEE High Performance Extreme Computing Conference (HPEC)*, pages 1–9. IEEE, 2023.

[99] John Schulman, Barret Zoph, Christina Kim, Jacob Hilton, Jacob Menick, Jiayi Weng, Juan Felipe Ceron Uribe, Liam Fedus, Luke Metz, Michael Pokorny, et al. Chatgpt: Optimizing language models for dialogue. *OpenAI blog*, 2(4), 2022.

[100] Jay Shah, Ganesh Bikshandi, Ying Zhang, Vijay Thakkar, Pradeep Ramani, and Tri Dao. Flashattention-3: Fast and accurate attention with asynchrony and low-precision. *arXiv preprint arXiv:2407.08608*, 2024.

[101] Weiyan Shi, Ryan Shea, Si Chen, Chiyuan Zhang, Ruoxi Jia, and Zhou Yu. Just fine-tune twice: Selective differential privacy for large language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 6327–6340, 2022.

[102] Zhenmei Shi, Junyi Wei, Zhuoyan Xu, and Yingyu Liang. Why larger language models do in-context learning differently? *arXiv preprint arXiv:2405.19592*, 2024.

[103] Tanmay Singh, Harshvardhan Aditya, Vijay K Madisetti, and Arshdeep Bahga. Whispered tuning: Data privacy preservation in fine-tuning llms through differential privacy. *Journal of Software Engineering and Applications*, 17(1):1–22, 2024.

[104] Charlie Snell, Ruiqi Zhong, Dan Klein, and Jacob Steinhardt. Approximating how single head attention learns. *arXiv preprint arXiv:2103.07601*, 2021.

[105] Mitchell Stern, Noam Shazeer, and Jakob Uszkoreit. Blockwise parallel decoding for deep autoregressive models. *Advances in Neural Information Processing Systems*, 31, 2018.

[106] Jovan Stojkovic, Esha Choukse, Chaojie Zhang, Inigo Goiri, and Josep Torrellas. Towards greener llms: Bringing energy-efficiency to the forefront of llm inference. *arXiv preprint arXiv:2403.20306*, 2024.

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

[108] Hanshi Sun, Zhuoming Chen, Xinyu Yang, Yuandong Tian, and Beidi Chen. Triforce: Lossless acceleration of long sequence generation with hierarchical speculative decoding. *arXiv preprint arXiv:2404.11912*, 2024.

[109] Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. A simple and effective pruning approach for large language models. In *The Twelfth International Conference on Learning Representations*, 2024.

[110] Hidenori Tanaka, Daniel Kunin, Daniel L Yamins, and Surya Ganguli. Pruning neural networks without any data by iteratively conserving synaptic flow. *Advances in neural information processing systems*, 33:6377–6389, 2020.

[111] Hans Thisanke, Chamli Deshan, Kavindu Chamith, Sachith Seneviratne, Rajith Vidanaarachchi, and Damayanthi Herath. Semantic segmentation using vision transformers: A survey. *Engineering Applications of Artificial Intelligence*, 126:106669, 2023.

[112] 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*, 2023.

[113] Szymon Tworkowski, Konrad Staniszewski, Mikołaj Pacek, Yuhuai Wu, Henryk Michalewski, and Piotr Miłoś. Focused transformer: Contrastive training for context scaling. *Advances in Neural Information Processing Systems*, 36, 2024.

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

[115] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[116] Chaoqi Wang, Guodong Zhang, and Roger Grosse. Picking winning tickets before training by preserving gradient flow. In *International Conference on Learning Representations*, 2019.

[117] Jiayu Wang, Yifei Ming, Zhenmei Shi, Vibhav Vineet, Xin Wang, and Neel Joshi. Is a picture worth a thousand words? delving into spatial reasoning for vision language models. *arXiv preprint arXiv:2406.14852*, 2024.

[118] Yilin Wang, Zeyuan Chen, Liangjun Zhong, Zheng Ding, Zhizhou Sha, and Zhuowen Tu. Dolfin: Diffusion layout transformers without autoencoder. *arXiv preprint arXiv:2310.16305*, 2023.

[119] Yilin Wang, Haiyang Xu, Xiang Zhang, Zeyuan Chen, Zhizhou Sha, Zirui Wang, and Zhuowen Tu. Omnicontrolnet: Dual-stage integration for conditional image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7436–7448, 2024.

[120] Yuntao Wang, Zirui Cheng, Xin Yi, Yan Kong, Xueyang Wang, Xuhai Xu, Yukang Yan, Chun Yu, Shwetak Patel, and Yuanchun Shi. Modeling the trade-off of privacy preservation and activity recognition on low-resolution images. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, pages 1–15, 2023.

[121] Zirui Wang, Zhizhou Sha, Zheng Ding, Yilin Wang, and Zhuowen Tu. Tokencompose: Grounding diffusion with token-level supervision. *arXiv preprint arXiv:2312.03626*, 2023.

[122] Dennis Wu, Jerry Yao-Chieh Hu, Teng-Yun Hsiao, and Han Liu. Uniform memory retrieval with larger capacity for modern hopfield models. In *Forty-first International Conference on Machine Learning (ICML)*, 2024.

[123] Dennis Wu, Jerry Yao-Chieh Hu, Weijian Li, Bo-Yu Chen, and Han Liu. STanhop: Sparse tandem hopfield model for memory-enhanced time series prediction. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024.

[124] Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, et al. The rise and potential of large language model based agents: A survey. *arXiv preprint arXiv:2309.07864*, 2023.

[125] Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. *arXiv preprint arXiv:2309.17453*, 2023.

[126] Chenwei Xu, Yu-Chao Huang, Jerry Yao-Chieh Hu, Weijian Li, Ammar Gilani, Hsi-Sheng Goan, and Han Liu. Bishop: Bi-directional cellular learning for tabular data with generalized sparse modern hopfield model. In *Forty-first International Conference on Machine Learning (ICML)*, 2024.

[127] Hu Xu, Gargi Ghosh, Po-Yao Huang, Prahal Arora, Masoumeh Aminzadeh, Christoph Feichtenhofer, Florian Metze, and Luke Zettlemoyer. Vlm: Task-agnostic video-language model pre-training for video understanding. *arXiv preprint arXiv:2105.09996*, 2021.

[128] Jing Xu, Arthur Szlam, and Jason Weston. Beyond goldfish memory: Long-term open-domain conversation. *arXiv preprint arXiv:2107.07567*, 2021.

[129] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057. PMLR, 2015.

[130] Xinchao Xu, Zhibin Gou, Wenquan Wu, Zheng-Yu Niu, Hua Wu, Haifeng Wang, and Shihang Wang. Long time no see! open-domain conversation with long-term persona memory. *arXiv preprint arXiv:2203.05797*, 2022.

[131] Yuhui Xu, Zhanming Jie, Hanze Dong, Lei Wang, Xudong Lu, Aojun Zhou, Amrita Saha, Caiming Xiong, and Doyen Sahoo. Think: Thinner key cache by query-driven pruning. *arXiv preprint arXiv:2407.21018*, 2024.

[132] Zhuoyan Xu, Zhenmei Shi, and Yingyu Liang. Do large language models have compositional ability? an investigation into limitations and scalability. In *ICLR 2024 Workshop on Mathematical and Empirical Understanding of Foundation Models*, 2024.

[133] Amir Zandieh, Insu Han, Majid Daliri, and Amin Karbasi. Kdeformer: Accelerating transformers via kernel density estimation. In *International Conference on Machine Learning*, pages 40605–40623. PMLR, 2023.

[134] Bowen Zhang, Zhi Tian, Quan Tang, Xiangxiang Chu, Xiaolin Wei, Chunhua Shen, et al. Segvit: Semantic segmentation with plain vision transformers. *Advances in Neural Information Processing Systems*, 35:4971–4982, 2022.

[135] Jieyu Zhang, Ranjay Krishna, Ahmed H Awadallah, and Chi Wang. Ecoassistant: Using llm assistant more affordably and accurately. *arXiv preprint arXiv:2310.03046*, 2023.

[136] Jingyi Zhang, Jiaxing Huang, Sheng Jin, and Shijian Lu. Vision-language models for vision tasks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[137] Jingzhao Zhang, Sai Praneeth Karimireddy, Andreas Veit, Seungyeon Kim, Sashank Reddi, Sanjiv Kumar, and Suvrit Sra. Why are adaptive methods good for attention models? *Advances in Neural Information Processing Systems*, 33:15383–15393, 2020.

[138] Michael Zhang, Kush Bhatia, Hermann Kumbong, and Christopher Ré. The hedgehog & the porcupine: Expressive linear attentions with softmax mimicry. *arXiv preprint arXiv:2402.04347*, 2024.

[139] Tianyi Zhang, Faisal Ladhak, Esin Durmus, Percy Liang, Kathleen McKeown, and Tatsunori B Hashimoto. Benchmarking large language models for news summarization. *Transactions of the Association for Computational Linguistics*, 12:39–57, 2024.

[140] Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, et al. H2o: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in Neural Information Processing Systems*, 36, 2024.

[141] Zixiao Zhang, Xiaoqiang Lu, Guojin Cao, Yuting Yang, Licheng Jiao, and Fang Liu. Vit-yolo: Transformer-based yolo for object detection. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 2799–2808, 2021.

[142] Chenyang Zhao, Xueying Jia, Vijay Viswanathan, Tongshuang Wu, and Graham Neubig. Self-guide: Better task-specific instruction following via self-synthetic finetuning. *arXiv preprint arXiv:2407.12874*, 2024.

# Appendix

**Roadmap.** In Section A, we offer the related work. In Section B, we provide a remark for our definition of loss function $L(X)$, showing that the most popular cross-entropy loss function is a special case of our general loss function definition. In Section C, we explain the techniques we employ, including low-rank approximation, some tricks for accelerating computation of gradients, and an analysis of the approximation error. In Section D, we discuss more extensions our algorithm, where we show that our algorithm facilitates the prompt tuning of LLMs, and preserves the potential of combining with concurrent hardware acceleration techniques. In Section E, we provide a detailed discussion about several potential extensions of our framework. In Section F, we introduce basic notations and concepts used in our paper, along with the low-rank approximation technique introduced in Alman and Song [3] and Alman and Song [4]. In Section G, we provide details about how we integrate the gradient of $T_i(X)$ into matrix form. In Section H, we explain how to apply the low-rank approximation technique to accelerate the computation for the gradient on $T_i(X)$. In Section I, we extend the result of Alman and Song [4] to arbitrary loss functions and accelerate the computation of gradient on $W$ via the low-rank approximation technique. In Section J, we calculate the gradient on $W_V$ and accelerate the computation of the gradient on $W_V$. In Section K, with the help of math induction, we analyze the time complexity and the approximation error across the entire model. In Section L, we discuss how our framework can expand to an attention mechanism with a causal attention mask. In Section M, we provide details about how to integrate our framework with attention mechanism with the residual connection. In Section N, we argue that, with the addition of multi-head attention, our algorithm can still achieve almost linear time gradient computation.

## Appendix A. Related Work

**Beyond the Previous Works** Our algorithm exhibits significant advancements over two brilliant related prior studies, namely Alman and Song [3] and Alman and Song [4]. In Alman and Song [3], the authors proposed an almost linear time algorithm for computing the forward process of the attention mechanism. In contrast, Alman and Song [4] introduced an almost linear time algorithm for the backward of attention mechanism. However, Alman and Song [4] has the following limitations: (1) only computing gradients for a single layer of the attention mechanism, which cannot extend to multiple layers; (2) calculating gradients with respect to a specific loss, namely the $\ell_2$ loss; (3) computing gradients only for the weight matrix $W_i$ (as defined in Definition 2), but ignore other crucial components such as the MLP layer following attention computation and the activation function. In our work, we have the following improvements beyond Alman and Song [4]: (1) we enable almost linear time gradient computation across an entire transformer layer, incorporating both the MLP layer and the activation function; (2) our algorithm supports gradient calculation for any loss function $L(X)$ (see Definition 5); (3) we extend the gradient calculation to include not only $W_i$ but also $T_i(X)$ and $W_{V_i}$. These advancements collectively demonstrate a substantial leap forward from the methodologies in Alman and Song [3] and Alman and Song [4].

**Attention mechanism.** Attention mechanisms, including self-attention and cross-attention, are pivotal techniques employed in state-of-the-art neural networks. Since it was introduced in Vaswani et al. [114], it has gained widespread adoption across various domains. In particular, it is integral to decoder-only LLMs [96] and the Vision Transformer (ViT) architecture [26]. The former has been instrumental in the remarkable success of LLMs, while the latter has significantly advanced the

**Algorithm 1** Almost Linear Time (ALT) Multi-layer Transformer Gradient Approximation Algorithm

**datastructure** ALTGRAD ▷ Theorem 8 and 9
 **members**
  $n \in \mathbb{R}$: the length of input sequence
  $d \in \mathbb{R}$: the hidden dimension
  $m \in \mathbb{R}$: the number of transformer layers
  $L(X) \in \mathbb{R}$: the loss function ▷ Definition 5
  $T_i \in \mathbb{R}^{n \times d}$: the output of $i$-th transformer layer
  $\mathsf{Attn}_i \in \mathbb{R}^{n \times d}$: the output that pass $i$-th attention layer
  $W_i, W_{V_i} \in \mathbb{R}^{d \times d}$ : the weight matrices in $i$-th transformer layer
 **end members**

**procedure** SINGLEGRAD($\frac{\mathrm{d}L(X)}{\mathrm{d}T_i}$) ▷ Theorem 8
  Compute $G_i = \frac{\mathrm{d}L(X)}{\mathrm{d}\mathsf{Attn}_i}$ via Lemma 14 ▷ $n^{1+o(1)}$ time
  Compute $\widetilde{D}_6, \widetilde{D}_7, \widetilde{D}_8, \widetilde{D}_2, \widetilde{D}_4$ via Lemma 55, 56, 58, 60 ▷ $n^{1+o(1)}$ time
  /* Approximate $\frac{\mathrm{d}L(X)}{\mathrm{d}T_{i-1}}$, Lemma 11 */
  $\widetilde{g}_t \leftarrow \widetilde{D}_6 + \widetilde{D}_7 + \widetilde{D}_8 + \widetilde{D}_2 + \widetilde{D}_4$ ▷ $n^{1+o(1)}$ time
  /* Approximate $\frac{\mathrm{d}L(X)}{\mathrm{d}W_i}$, Lemma 12 */
  Construct $U_3, V_3$ via Lemma 12 ▷ $n^{1+o(1)}$ time
  $\widetilde{g}_w \leftarrow (T_{i-1}^\top U_3) \cdot (V_3^\top T_{i-1})$ ▷ $n^{1+o(1)}$ time
  /* Approximate $\frac{\mathrm{d}L(X)}{\mathrm{d}W_{V_i}}$, Lemma 13 */
  Construct $U_1, V_1$ via Lemma 27 ▷ $n^{1+o(1)}$ time
  $\widetilde{g}_v \leftarrow (T_{i-1}^\top U_1) \cdot (V_1^\top G_i)$ ▷ $n^{1+o(1)}$ time
  **return** $\widetilde{g}_t, \widetilde{g}_w, \widetilde{g}_v$ ▷ $\widetilde{g}_t, \widetilde{g}_w, \widetilde{g}_v$ are the approximated $\frac{\mathrm{d}L(X)}{\mathrm{d}T_{i-1}}, \frac{\mathrm{d}L(X)}{\mathrm{d}W_i}, \frac{\mathrm{d}L(X)}{\mathrm{d}W_{V_i}}$ in next iteration, respectively.
**end procedure**

**procedure** MULTIGRAD($L(X)$) ▷ Theorem 9
  Compute $\frac{\mathrm{d}L(X)}{\mathrm{d}T_m}$ ▷ $O(nd)$ time
  $\widetilde{g}_t \leftarrow \frac{\mathrm{d}L(X)}{\mathrm{d}T_m}$
  **for** $i = m \to 1$ **do**
   $\widetilde{g}_t, \widetilde{g}_w, \widetilde{g}_v \leftarrow$ SINGLEGRAD ($\widetilde{g}_t$)
   Optimize $W_i$ via $\widetilde{g}_w$ using optimizer
   Optimize $W_{V_i}$ via $\widetilde{g}_v$ using optimizer
  **end for**
**end procedure**
 **end datastructure**

field of computer vision, encompassing applications such as image generation [97, 119, 121], detection [68, 141], segmentation [111, 134], and layout generation [14, 40, 118]. Moreover, attention mechanism can be integrated into multi-modal models [77, 117, 127, 136], math reasoning [62], diffusion models [28, 50, 75, 87, 94], differential privacy [10, 76, 101, 103, 120] and many other techniques.

**Long-context modeling in LLMs.** As LLMs grow in size and capability, in-context learning (ICL) [91, 102, 132] has become a preferred method for directing these models to perform a variety of tasks, as opposed to the resource-intensive process of fine-tuning. Nonetheless, research has indicated that longer prompts can impair LLMs performance due to the limitation on maximum sequence length during pre-training [65]. Consequently, extending the maximum sequence length during pre-training and fine-tuning stages is imperative. Enhancing training efficiency is crucial given the prevalent use of the Transformer architecture in LLMs, which incurs a quadratic computational cost relative to sequence length. Addressing this challenge, some studies have explored continued fine-tuning of LLMs with extended context lengths [113], while others have experimented with the interpolation and extrapolation capabilities of positional embedding [16, 95, 107]. However, these approaches have not fundamentally addressed the core issue: the quadratic computational cost associated with sequence length in the attention mechanism [30, 55]. In this study, we delve into accelerating the attention mechanism, thereby addressing the long-context modeling issue at its essence.

**Attention acceleration.** Attention mechanism has faced criticism due to its quadratic time complexity with respect to context length, a concern exacerbated by the increasing length in modern large language models (LLMs) such as GPT-4 [1], Claude 3.5 [7], Llama 3.1 [83, 112], etc. Nevertheless, this limitation can be circumvented by employing polynomial kernel approximation techniques [2], which enable the derivation of a low-rank representation of the attention matrix. This innovation significantly accelerates both the training and inference processes of a single attention layer, achieving almost linear time complexity [3, 4], while our work supports both training and inference for any multi-layer transformer. Furthermore, this approach can be extended to higher-order attention mechanisms, i.e., tensor attention, maintaining almost linear time complexity during both training and inference [5, 77]. Moreover, there are other theoretical approaches. For instance, Liang et al. [72] introduces the conv-basis method to accelerate attention computation. Han et al. [41] proposes a near-linear time algorithm under the assumptions of uniform softmax column norms and sparsity.

Other approaches involve modifying model architectures to enable faster inference, such as Mamba [21, 39], Linearizing Transformers [90, 138], PolySketchFormer [54], and the Hopfield Model [45–48, 122, 123, 126] and so on. Another line of work is to prune the weights in a neural network to reduce running time and memory consumption [9, 11, 17, 31–33, 42, 44, 52, 53, 58, 67, 79, 109, 110, 116]. In addition, specific techniques have been developed to accelerate LLM generation, including KV-Cache compression [25, 38, 70, 82, 125, 131, 140] and speculative decoding [27, 64, 108].

**Attention theory.** Bahdanau et al. [8] introduced attention mechanisms in NLP, enhancing encoder-decoder architecture with variable-length vectors to improve machine translation. Building on this, Luong et al. [84] developed local and global attention variants, further refining NLP tasks. Attention mechanisms found diverse applications: Xu et al. [129] applied them to image captioning, Vaswani et al. [114]'s Transformer model revolutionized NLP by capturing word relationships, and Veličković et al. [115] incorporated attention into graph neural networks. Recent Large Language Model research has focused extensively on attention computation [3, 12, 15, 23, 56, 71, 133]. Studies by Chen et al. [15], Kitaev et al. [56], Zandieh et al. [133] use Locality Sensitive Hashing for attention approximation, with Zandieh et al. [133] offering efficient dot-product attention. Brand et al. [12] and Alman and Song [3] explore static and dynamic attention calculations, while Li et al. [71] investigates hyperbolic regression regularization. Deng et al. [23] proposes algorithms for reducing attention matrix dimensionality in LLMs. Attention has also been examined from optimization and convergence perspectives [34, 69, 104, 137], investigating word co-occurrence learning [69], regression problems with exponential activation functions [34], attention mechanism evolution during training [104], and the impact of heavy-tailed noise on stochastic gradient descent [137]. Theoretical explorations of attention variants include quantum attention [35], tensor attention [5, 77], and differentially private attention [36, 73, 76].

## Appendix B. Remark of Loss Function Definition

**Remark 10** *Typically, in Definition 5, the most widely used loss function in the LLM training procedure is the cross-entropy loss function, which can also be viewed as a summation of one unit loss function.*

*The output matrix of the multi-layer transformer needs to pass an additional linear layer to map the hidden dimension* $d$ *to the vocabulary size* $d_{\mathrm{voc}}$*. Assuming* $d_{\mathrm{voc}}$ *is a constant, the weight matrix dimensions for this additional MLP layer are* $d \times d_{\mathrm{voc}}$*. The probability tensor* $Y_{\mathrm{pred}} \in \mathbb{R}^{n \times d_{\mathrm{voc}}}$ *is the final output. We denote the ground truth as* $Y_{\mathrm{gt}} \in \mathbb{R}^{n \times d_{\mathrm{voc}}}$ *corresponding to* $Y_{\mathrm{pred}}$*. According to the cross-entropy loss definition, the formula is expressed as*

$$L_{\mathrm{cross-entropy}}(X) = -\sum_{j=1}^{n} \sum_{k=1}^{d_{\mathrm{voc}}} (Y_{\mathrm{gt}})_{j,k} \log((Y_{\mathrm{pred}})_{j,k})$$

*where the summation iterates over all elements, and the ground truth* $(Y_{\mathrm{gt}})_{j,k} = 1$ *for the correct class and* $0$ *otherwise.*

## Appendix C. Technical Overview

### C.1. Low-rank approximation for attention matrix

In this section, we delve into the crucial technique behind our work: the low-rank approximation of the attention matrix, which is achieved through the polynomial method [2, 6]. Drawing inspiration from Alman and Song [3], the intuition of this approximation lies in the fact that the attention matrix $f(X) \in \mathbb{R}^{n \times n}$ (as defined in Definition 2), also referred to as the similarity matrix in attention mechanism, can be effectively approximated by low-rank matrices $U_1, V_1 \in \mathbb{R}^{n \times k_1}$, where $k_1 = n^{o(1)}$. The naive method for calculating the attention matrix $f(X)$ has a time complexity of $O(n^2)$, whereas the input data $X \in \mathbb{R}^{n \times d}$ contains only $d \cdot n = n^{1+o(1)}$ entries. This discrepancy suggests the potential of using low-rank representations of $f(X)$ to design a fast algorithm.

An example of how to use the low-rank representations is the attention forward ($\mathsf{Attn}(X) := f(X)V$ in Definition 2) as in Alman and Song [3]: approximating $f(X)$ along does not lead to fast algorithm, since $U_1 V_1^\top$ still requires $n \times n$ entries. But by using the structure of the whole, we can do it faster. By expressing $f(X)$ as $U_1 V_1^\top$, the attention forward becomes $\underbrace{U_1}_{n \times k_1} \underbrace{V_1^\top}_{k_1 \times n} \underbrace{V}_{n \times d}$. It is well known that different multiplication sequences can lead to dramatically different numbers of operations required, so the order of matrix multiplications matters. We first perform $V_1^\top V \in \mathbb{R}^{k_1 \times d}$ and this cost $O(k_1 n d) = n^{1+o(1)}$ time. Then we perform $U_1 V_1^\top V$ costing $O(n k_1 d) = n^{1+o(1)}$ time.

This method significantly reduces the computation time of the attention forward from $O(n^2)$ to almost linear time, $n^{1+o(1)}$. Driven by this technique and analyzing the close forms of the gradients, we can extend the acceleration to the gradient of the entire model.

### C.2. Accelerating gradient computation of $T_i(X)$

Based on the low-rank approximation method mentioned in Section C.1, we can compute the gradient of $L(X)$ with respect to the intermediate variable $T_i(X)$, which denotes the output of the $i$-th transformer layer. This computation is critical as it enables us to calculate gradients for other gradient components because of the chain rule.

**Extending to any kind of loss function.** According to the findings in Deng et al. [24], the gradient $\frac{\mathrm{d}L(X)}{\mathrm{d}T_i(X)}$ can be decomposed into five components, namely $C_6(X), C_7(X), C_8(X), C_2(X), C_4(X)$, as detailed in Lemma 34. However, the gradient result presented in Deng et al. [24] is tailored to a specific loss function, the $\ell_2$ loss, limiting its applicability to a narrow range of scenarios. In this study, we extend the scope of their findings by extending them to apply to any loss function $L(X)$, as defined in Definition 5. By incorporating $G_i = \frac{\mathrm{d}L(X)}{\mathrm{d}\mathsf{Attn}_i(T_{i-1}(X))}$, we derive a closed-form expression for the gradient of $L(X)$ with respect to $T_i(X)$, which is detailed in Section G.2.

**Accelerating the gradient computation.** To accelerate the gradient computation for $T_i(X)$, we need the matrix form of the gradients, as discussed in Section G. This approach is essential for understanding the underlying mechanisms when applying the low-rank approximation technique in gradient calculations. Subsequently, using that technique, we can accelerate the gradient computation for $\frac{\mathrm{d}L(X)}{\mathrm{d}T_i(X)}$ (Lemma 11). By individually applying this technique to each of the five terms, we demonstrate that each term can be computed in almost linear time, $n^{1+o(1)}$, as shown in Sections H.1, H.2, H.3, H.4, and H.5.

The next step is to aggregate these terms, as described in Section H.6. Since all five terms are $n \times d$ matrices, the summation of these terms remains almost linear in complexity. Conclusively, we are safe to argue that for any single-layer transformer, the gradient computation with respect to the input tensor can be performed in almost linear time $n^{1+o(1)}$, as stated in Lemma 11.

The statement made for a single transformer layer can be readily generalized to any layer within an $m$-layer transformer model. For instance, consider the intermediate variables $T_i(X)$ and $T_{i-1}(X)$ (as defined in Definition 6), where $T_i(X) = (g_i \circ \mathsf{Attn}_i)(T_{i-1}(X))$. Given the gradient $\frac{\mathrm{d}L(X)}{\mathrm{d}T_i(X)}$, as established in the previous paragraph, we can compute the gradient with respect to $T_{i-1}(X)$, namely $\frac{\mathrm{d}L(X)}{\mathrm{d}T_{i-1}(X)}$, in almost linear time $n^{1+o(1)}$. For a multi-layer transformer model, the above process can be conducted recursively. Thus, we can compute the gradient of the loss function $L(X)$ on any $T_i(X)$ in almost linear time $n^{1+o(1)}$.

**Lemma 11 (Fast computation for $\frac{\mathrm{d}L(X)}{\mathrm{d}T_i(X)}$, informal version of Lemma 61)** *Let $L(X)$ be defined as Definition 5. Let $m$ denote the number of self-attention transformer layers (see Definition 3). Let $T_i(X)$ denote the intermediate variable output by $i$-th self-attention transformer layer (see Definition 6). We show that $\frac{\mathrm{d}L(X)}{\mathrm{d}T_i(X)}$ can be approximated in $n^{1+o(1)}$ time, with $1/\operatorname{poly}(n)$ approximation error.*

### C.3. Accelerating gradient computation of $W_i$ and $W_{V_i}$

In Section C.2, we detailed the fast computation of gradients for intermediate variables $T_i(X)$. Given that $W_i$ is defined as the product $W_{Q_i} W_{K_i}^\top$ (see Definition 2), with $W_{Q_i}$ and $W_{K_i}$ representing the query and key weight matrices, respectively, the gradients of $W_i$ and $W_{V_i}$ represent *all* trainable weight matrices in a transformer layer. Consequently, by determining the gradients for $W_i$ and $W_{V_i}$ across each layer, we achieve almost linear time gradient back-propagation throughout multi-layer transformer models.

**Fast gradient computation.** The prior study in Alman and Song [4] demonstrated that the gradient of $W_i$ can be computed in almost linear time. We extend their findings by adapting their approach to accommodate any loss function $L(X)$ (as defined in Definition 5) and further generalize their results to include the gradient computation for both $W_i$ and $W_{V_i}$ in each transformer layer (Lemma 12 and 13).

**Lemma 12 (Fast computation for $\frac{\mathrm{d}L(X)}{\mathrm{d}W_i}$, informal version of Lemma 66)** *Let $L(X)$ be defined as Definition 5, and $m$ be the number of self-attention transformer layers (Definition 3). For any $i \in [m]$, let $W_i = W_{Q_i} W_{K_i}^\top, W_{V_i} \in \mathbb{R}^{d \times d}$ denote the attention weight in the $i$-th transformer layer. We show that $\frac{\mathrm{d}L(X)}{\mathrm{d}W_i}$ can be approximated in $n^{1+o(1)}$ time, with $1/\operatorname{poly}(n)$ approximation error.*

**Lemma 13 (Fast computation for $\frac{\mathrm{d}L(X)}{\mathrm{d}W_{V_i}}$, informal version of Lemma 70)** *Let $L(X)$ be defined as Definition 5, and $m$ be the number of self-attention transformer layers (Definition 3). For any $i \in [m]$, let $W_i = W_{Q_i} W_{K_i}^\top, W_{V_i} \in \mathbb{R}^{d \times d}$ denote the attention weight in the $i$-th transformer layer. We show that $\frac{\mathrm{d}L(X)}{\mathrm{d}W_{V_i}}$ can be approximated in $n^{1+o(1)}$ time, with $1/\operatorname{poly}(n)$ approximation error.*

### C.4. Accelerating gradient computation for multi-Layer transformers

In this section, our focus turns to extending the single-layer transformer result from the previous section to a multi-layer transformer.

**Running time analysis.** We derive the closed-form gradient for the non-attention components within a transformer layer, namely the $g_i$ function defined in Definition 3. With the closed-form gradient of $g_i$ established in Lemma 71, we then demonstrate in Lemma 14 that the gradient computation for $g_i$ can also be achieved in almost linear time. Given that the number of layers $m$ is much smaller than $n$, we can treat $m$ as a constant. Consequently, with respect to running time, since the computation time for gradients on each layer is $n^{1+o(1)}$, we only need to iteratively repeat this procedure for $m$ time. Therefore, the overall running time for computing gradients across the entire model is $m \cdot n^{1+o(1)}$.

**Lemma 14 (Computation time for $G_i$, informal version of Lemma 72)** *Let $T_i(X)$ be defined as Definition 6, i.e. $T_i(X) = (g_i \circ \mathsf{Attn}_i)(T_{i-1}(X))$. Let $G_i \in \mathbb{R}^{n \times d}$ denote the gradient matrix resulting from the application of the chain rule up to the function $g_i$, i.e., $G_i = \frac{\mathrm{d}L(X)}{\mathrm{d}\mathsf{Attn}_i(T_{i-1}(X))}$. Assume we already have $\frac{\mathrm{d}L(X)}{\mathrm{d}T_i(X)}$. Assuming for any $Z \in \mathbb{R}^{n \times d}$, we have $g_i(Z) \in \mathbb{R}^{n \times d}$, and $g_i(Z) = \phi(Z \cdot W_g)$, where $W_g \in \mathbb{R}^{d \times d}$ and $\phi : \mathbb{R} \to \mathbb{R}$ denotes any element-wise activation function. Let $\phi'$ denote the derivative of $\phi$. Then, we show that $G_i$ can be computed in $n^{1+o(1)}$ time.*

**Error propagation analysis.** Here, we consider the approximation error. In our setting, the approximation error originates from the low-rank approximation of the attention matrix, as detailed in Lemma 27. As discussed in previous sections, the approximation error in each layer can be bounded by $1/\operatorname{poly}(n)$. Then, we only need to focus on how error propagates in different layers.

We first prove that our $1/\operatorname{poly}(n)$ approximation error statement holds for a single-layer transformer, as evidenced in Lemma 73. Subsequently, through mathematical induction and leveraging the results of error propagation over the gradient of $g_i$, we can show that the approximation error can be bounded by $1/\operatorname{poly}(n)$ for any $m$-layer transformer (Lemma 15), where the number of layers $m$ is considered as a constant.

**Lemma 15 (Multi-layer transformer gradient approximation, informal version of Theorem 74)** *Let $L(X)$ be defined as Definition 5. Let $X$ be defined as Definition 2. Suppose we have a $m$-layer transformer (see Definition 3). Then, for any $i \in [m]$, we can show that: $(i)$ Running time: Our algorithm can approximate $\frac{\mathrm{d}L(X)}{\mathrm{d}T_{i-1}(X)}$, $\frac{\mathrm{d}L(X)}{\mathrm{d}W_i}$, and $\frac{\mathrm{d}L(X)}{\mathrm{d}W_{V_i}}$ in $n^{1+o(1)}$ time; $(ii)$ Error bound: The approximation of the entire transformer model can be bounded by $1/\operatorname{poly}(n)$. Namely, our algorithm output $\widetilde{g}$ satisfies $\|\widetilde{g} - \frac{\mathrm{d}L(X)}{\mathrm{d}X}\|_\infty \leq 1/\operatorname{poly}(n)$.*

## Appendix D. More Extensions

**Prompt tuning.** Prompt tuning (or prefix learning) is a prevalent approach in parameter-efficient fine-tuning (PEFT), which requires the calculation of gradients on input data $X$. Given our algorithm's ability to compute gradients for intermediate variables $T_i$ in approximately linear time, we can similarly accelerate the gradient computation for input data $X$, thus enhancing the efficiency of the prompt tuning process. Additional details are provided in Section E.5.

**Synergy with system-level attention acceleration.** Many contemporary works focus on system-level acceleration of attention mechanisms, often by leveraging caching and mitigating I/O bottlenecks. Our algorithm has the potential to integrate with such advancements. By combining our theoretical improvements in computation time (from $O(n^2)$ to $n^{1+o(1)}$) with system-level optimizations, the overall efficiency of attention mechanism computation may increase dramatically. We leave the implementation of our method on GPU as future work since there are several coding challenges. More details can be found in Section E.4.

## Appendix E. Discussion and Extension Details

In Section E.1, we argue that our framework can easily adapt to the multi-head attention mechanism. In Section E.2, we introduce how to integrate residual connection to our framework. In Section E.3, we detail the integration of the causal attention mask into our algorithm. In Section E.4, we discuss the possibility of the synergy between our theoretical side attention acceleration and the existing system-level attention acceleration mechanism. In Section E.5, we show how to expedite prompt tuning using our results.

### E.1. Multi-head attention

The multi-head attention mechanism was first introduced by Vaswani et al. [114]. This innovation allows a token to simultaneously attend to multiple positions within the same layer, thereby enriching the model's capacity for capturing various dependencies. However, this enhanced capability comes with an increase in the size of the attention matrix $f(X)$ from $1 \times n \times n$ to $h \times n \times n$, where $h$ is the number of attention heads. To mitigate the computational burden, each head's vector is derived by splitting the original vector, reducing the dimensionality of each head to $d_h := d/h$. To summarize, the key distinctions between multi-head and single-head attention are (1) an enlarged attention matrix $f(X)$ and (2) a reduced dimensionality $d_h$ within each attention head.

**Enlarged attention matrix.** As previously discussed, the attention matrix's dimensionality increases with the number of heads, $h$. Despite this expansion, the application of the low-rank approximation technique, as outlined in Section C.1, ensures that the computation time for the attention matrix remains almost linear. Specifically, for a constant number of heads $h$ in the multi-head mechanism, the time complexity for computing $f(X) \in \mathbb{R}^{h \times n \times n}$ is $h \cdot n^{1+o(1)} = n^{1+o(1)}$.

**Reduced dimensionality.** Another differentiating factor of multi-head attention is the lower dimensionality processed by each head, i.e. $d_h := d/h$, compared the full $d$ in single-head attention. This reduction ensures that the gradient computation time does not increase with the introduction of multiple attention heads.

We provide comprehensive analysis of the synergy of our algorithm with multi-head attention in Section N. We first prove in Lemma 89, with the addition of multi-head attention, the gradient over the attention mechanism can be computed in almost linear time. Then, we further prove that for any multi-layer transformer, with multi-head attention, the gradient can be computed in almost linear time as well.

### E.2. Residual connection

Residual connection is a pivotal technique in deep neural network architectures, effectively addressing issues such as vanishing and exploding gradients during training process, and facilitating faster convergence of the model. Residual connection is also integrated into the standard attention mechanism. Formally, given the intermediate variable $T_i(X)$ output by the $i$-th transformer layer as defined in Definition 6, we provide the formal definition of residual connection in Definition 83 and 84. Since the residual connection only brings an additional add operation to each component and with $T_i(X)$ belonging to the space $\mathbb{R}^{n \times d}$, the residual connection introduces only a marginal computational overhead of $O(n \cdot d)$ per layer. Consequently, the total computational cost for each layer is $O(n \cdot d) + n^{1+o(1)} = n^{1+o(1)}$. Hence, by intuition, the inclusion of residual connections does not compromise the overall complexity of our method.

The detailed analysis is provided in Section M, where we first prove in Lemma 85, that if the gradient over one structure can be computed in almost linear time, then with the addition of the residual connection, the gradient can also be computed in almost linear time. Then we use math induction to extend our result to the entire multi-layer transformer model.

### E.3. Causal attention mask

In transformer training, attention mask is a crucial component, designed to prevent a given token from attending to future tokens in the sequence. Causal attention mask is a widely used attention mask, which is configured as a lower triangular matrix, where elements on or below the main diagonal are ones, with all other entries being zeros.

Now we describe how to incorporate this into our algorithm. Let $M \in \{0,1\}^{n \times n}$ represent the causal attention mask (see Definition 76). Let $\widehat{f}(X) := D^{-1}(M \odot A)$ where $A = \exp(XWX^\top/d)$

and $D := \mathrm{diag}((M \odot A) \cdot \mathbf{1}_n)$. Lemma 75 reveals that $A$ has a low-rank representation given by $U_0 V_0^\top$. Using Lemma 77, we know $(M \odot (U_0 V_0^\top)) \cdot v$ for any vector $v \in \mathbb{R}^n$ can be computed in almost linear time.

To integrate the causal mask into the gradient computation within each transformer layer, we first find all instances that have the structure of $f(X) \cdot H$ or $(f(X) \odot (UV^\top)) \cdot H$, where $H, U, V$ are low rank matrices. Then, we replace $f(X)$ with $\widehat{f}(X)$ in these instances. More detailed analysis of causal attention can be found in Section L. To be more specific, we group the gradient components for $T_i, W_i, W_{V_i}$ into two categories, one for dot product (Lemma 81), another for Hadamard product (Lemma 82). After showing each component can be calculated in almost linear time, the overall gradient computation remains $n^{1+o(1)}$ time. Thus, our framework can seamlessly accommodate causal attention masks.

### E.4. System-level attention acceleration

The attention computing acceleration involves a two-pronged strategy that leverages both system-level improvements (e.g. Flash Attention [20, 22, 100]) and the theoretical time complexity improvements (e.g. our work and Han et al. [41]).

Numerous efforts have been made in the literature to accelerate attention calculations at the system level. For instance, Flash Attention [20, 22, 100] targets the I/O bottleneck inherent in attention mechanisms. Studies such as block-wise parallel decoding [105] focus on implementing parallel decoding within transformer models to enhance inference speed. Additionally, recent advancements in the field of speculative decoding, such as Medusa [13], leverage a smaller, more efficient model to generate predictions, with the larger model only responsible for validating, the smaller model's outputs [60].

Despite these innovations, the aforementioned methods do not address the fundamental quadratic time complexity $O(n^2)$ of the attention mechanisms. This presents an opportunity to complement our low-rank approximation technique, with these system-level optimizations, thereby achieving an even greater acceleration in attention computation. For instance, we could design an I/O-aware algorithm for Algorithm 1, similar to the approach taken by Flash Attention, to effectively leverage GPU acceleration.

To implement our algorithm practically on GPU, we have some coding challenges to fix: (1) we need to define some new tensor operations in PyTorch, e.g. Eq. (5), Eq. (8); (2) we need to systematically re-implement some back-propagation function of the current PyTorch function; (3) we need to implement some CUDA function to run our algorithm in parallel for the casual mask, see discussion in Section E.3. We may leave this as our future work.

### E.5. Prompt tuning

Prompt tuning, as introduced by various studies [49, 59, 66, 74, 80, 92], has emerged as a parameter-efficient fine-tuning strategy for large language models (LLMs). Specifically, prompt tuning involves adjusting "soft prompts" conditioned on frozen LLMs. This method requires relatively small number of tuneable parameters compared with fine-tuning the entire LLMs, making it a popular choice for conserving training resources, including data and computational power.

The analysis reveals that the essence of prompt tuning involves computing gradients with respect to the soft prompts $X_p$ across the entire model. In both prompt tuning and full fine-tuning, the quadratic $O(n^2)$ computational complexity of gradient calculation remains the same due to the self-attention mechanism inherent in LLMs.

In this work, leveraging the low-rank approximation technique discussed in Section C.1, our algorithm (Algorithm 1) efficiently computes gradients on soft prompts $X_p$ over the entire model in almost linear time. This suggests that our method is universal and can also be applied within traditional prompt tuning frameworks.

## Appendix F. Preliminary on Gradient Calculation

In Section F.1, we list several useful math facts used in the following sections of this paper. In Section F.2, we provide the close forms of the gradient components. In Section F.3, we introduce some mathematical definitions to facilitate understanding of gradient calculations. In Section F.4, we list some low rank approximation technique introduced in Alman and Song [3] and Alman and Song [4]. In Section F.5, we demonstrate that the entries of matrices defined in Section F.3 are bounded.

**Notations.** For two vectors $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^n$, we use $\langle x, y \rangle$ to denote the inner product between $x, y$. Namely, $\langle x, y \rangle = \sum_{i=1}^n x_i y_i$. We use $e_i$ to denote a vector where only $i$-th coordinate is 1, and other entries are 0. For each $a, b \in \mathbb{R}^n$, we use $a \odot b \in \mathbb{R}^n$ to denote the Hardamard product, i.e. the $i$-th entry of $(a \odot b)$ is $a_i b_i$ for all $i \in [n]$. We use $\mathbf{1}_n$ to denote a length-$n$ vector where all the entries are ones. We use $\|A\|_\infty$ to denote the $\ell_\infty$ norm of a matrix $A \in \mathbb{R}^{n \times d}$, i.e. $\|A\|_\infty := \max_{i \in [n], j \in [d]} |A_{i,j}|$. We use $\mathrm{poly}(n)$ to denote polynomial time complexity with respective to $n$.

### F.1. Basic math facts

In this section, we provide some useful basic math facts,

**Fact 16** *Let $x, y, z \in \mathbb{R}^n$. Then we have*

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

Then, we introduce a classical folklore used for the Hadamard product of two matrices.

**Fact 17 (Folklore, [4])** *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*

$$(\underbrace{U_1}_{n \times k_1} \underbrace{V_1^\top}_{k_1 \times n}) \odot (\underbrace{U_2}_{n \times k_2} \underbrace{V_2^\top}_{k_2 \times n}) = \underbrace{(U_1 \oslash U_2)}_{n \times k_1 k_2} \underbrace{(V_1 \oslash V_2)^\top}_{k_1 k_2 \times n}$$

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

### F.2. Close form of three gradient components

In this section, we show how to derive the close form for the gradient components within each layer of a multi-layer transformer.

**Lemma 18 (Close form of gradient components, formal version of Lemma 7)** *If we have the below conditions,*

- *Let $L(X)$ be defined as Definition 5.*
- *Let $W_i := W_{Q_i} W_{K_i}^\top \in \mathbb{R}^{d \times d}$ be the key-query weight matrix, $W_{V_i} \in \mathbb{R}^{d \times d}$ be the value weight matrix for the $i$-th transformer layer.*
- *Let $T_i(X)$ denote the intermediate variable output by $i$-th self-attention transformer layer (see Definition 6).*
- *Let $G_i \in \mathbb{R}^{n \times d}$ denote the gradient matrix resulting from the application of the chain rule up to the function $g_i$, i.e., $G_i = \frac{\mathrm{d}L(X)}{\mathrm{d}\mathsf{Attn}_i(T_{i-1}(X))}$.*
- *For $i_2 \in [n], j_2 \in [d]$, let $G_i(i_2, j_2)$ denote the $(i_2, j_2)$-th entry of $G_i$, let $\frac{\mathrm{d}\mathsf{Attn}_i(T_{i-1}(X))_{i_2,j_2}}{\mathrm{d}T_{i-1}(X)} \in \mathbb{R}^{n \times d}$ denote the gradient of $(i_2, j_2)$-th entry of $\mathsf{Attn}_i(T_{i-1}(X))$.*

*Then, we can show that*

- ***Part 1.***

$$\frac{\mathrm{d}L(X)}{\mathrm{d}T_{i-1}(X)} = \sum_{i_2=1}^{n} \sum_{j_2=1}^{d} G_i(i_2, j_2) \cdot \frac{\mathrm{d}\mathsf{Attn}_i(T_{i-1}(X))_{i_2,j_2}}{\mathrm{d}T_{i-1}(X)}.$$

- ***Part 2.***

$$\frac{\mathrm{d}L(X)}{\mathrm{d}W_i} = \sum_{i_2=1}^{n} \sum_{j_2=1}^{d} G_i(i_2, j_2) \cdot \frac{\mathrm{d}\mathsf{Attn}_i(T_{i-1}(X))_{i_2,j_2}}{\mathrm{d}W_i}.$$

- ***Part 3.***

$$\frac{\mathrm{d}L(X)}{\mathrm{d}W_{V_i}} = \sum_{i_2=1}^{n} \sum_{j_2=1}^{d} G_i(i_2, j_2) \cdot \frac{\mathrm{d}\mathsf{Attn}_i(T_{i-1}(X))_{i_2,j_2}}{\mathrm{d}W_{V_i}}.$$

**Proof** We have

- $L(X) \in \mathbb{R}$.
- $\mathsf{Attn}_i(T_{i-1}(X)) \in \mathbb{R}^{n \times d}, T_{i-1}(X) \in \mathbb{R}^{n \times d}$.
- $W_i \in \mathbb{R}^{d \times d}, W_{V_i} \in \mathbb{R}^{d \times d}$.

Therefore, we have

- $\frac{\mathrm{d}L(X)}{\mathrm{d}T_{i-1}(X)} \in \mathbb{R}^{n\times d}$, $\frac{\mathrm{d}\mathsf{Attn}_i(T_{i-1}(X))}{\mathrm{d}T_{i-1}(X)} \in \mathbb{R}^{(n\times d)\times(n\times d)}$.
- $\frac{\mathrm{d}L(X)}{\mathrm{d}W_i} \in \mathbb{R}^{d\times d}$, $\frac{\mathrm{d}\mathsf{Attn}_i(T_{i-1}(X))}{\mathrm{d}W_i} \in \mathbb{R}^{(n\times d)\times(d\times d)}$.
- $\frac{\mathrm{d}L(X)}{\mathrm{d}W_{V_i}} \in \mathbb{R}^{d\times d}$, $\frac{\mathrm{d}\mathsf{Attn}_i(T_{i-1}(X))}{\mathrm{d}W_{V_i}} \in \mathbb{R}^{(n\times d)\times(d\times d)}$.

Then, simply applying chain rule, we can get the final results. ■

### F.3. Basic notations for computing gradients

Before we move on to compute gradients, we need to define some useful notations.

We begin with introducing the index for a matrix.

**Definition 19 (Simplified notations)** *For any matrix $Z \in \mathbb{R}^{n\times d}$, for $i \in [n], j \in [d]$, we have following definitions:*

- *Let $\underbrace{Z_{i,j}}_{\text{scalar}}$ and $Z(i,j)$ denote the $(i,j)$-th entry of $Z$.*
- *Let $\underbrace{Z_{i,*}}_{d\times 1}$ and $Z(i,*)$ denote the $i$-th row of $Z$.*
- *Let $\underbrace{Z_{*,j}}_{n\times 1}$ and $Z(*,j)$ denote the $j$-th column of $Z$.*

Then, we define the exponential matrix in the attention mechanism.

**Definition 20 (Exponential function $u$)** *If we have the below conditions,*

- *Let $X \in \mathbb{R}^{n\times d}$*
- *Let $W := W_Q W_K^\top \in \mathbb{R}^{d\times d}$*

*We define $u(X) \in \mathbb{R}^{n\times n}$ as follows*

$$u(X) := \exp(XWX^\top)$$

Then, we introduce the summation vector of the aforementioned exponential matrix.

**Definition 21 (Sum function of softmax $\alpha$)** *If we have the below conditions,*

- *Let $X \in \mathbb{R}^{n\times d}$*
- *Let $u(X)$ be defined as Definition 20*

*We define $\alpha(X) \in \mathbb{R}^n$ as follows*

$$\alpha(X) := u(X) \cdot \mathbf{1}_n$$

Then, with the help of the summation vector, we are ready to normalize the exponential matrix and get the softmax probability matrix.

**Definition 22 (Softmax probability function $f$)** *If we have the below conditions,*

- *Let $X \in \mathbb{R}^{n \times d}$*
- *Let $u(X) \in \mathbb{R}^{n \times n}$ be defined as Definition 20*
- *Let $\alpha(X) \in \mathbb{R}^{n}$ be defined as Definition 21*

*We define $f(X) \in \mathbb{R}^{n \times n}$ as follows*

$$f(X) := \mathrm{diag}(\alpha(X))^{-1} u(X)$$

*where we define $f(X)_{j_0}^\top \in \mathbb{R}^n$ is the $j_0$-th row of $f(X)$.*

Besides the probability matrix introduced above, we introduce the value matrix in the following definition.

**Definition 23 (Value function $h$)** *If we have the below conditions,*

- *Let $X \in \mathbb{R}^{n \times d}$*
- *Let $W_V \in \mathbb{R}^{d \times d}$*

*We define $h(X) \in \mathbb{R}^{n \times d}$ as follows*

$$h(X) = X W_V$$

Then, we introduce $s(X)$ to represent the output of the attention mechanism.

**Definition 24 (Self-attention output $s$)** *If we have the below conditions,*

- *Let $f(X)$ be defined as Definition 22*
- *Let $h(X)$ be defined as Definition 23*

*We define $s(X) \in \mathbb{R}^{n \times d}$ as follows*

$$s(X) = f(X) h(X)$$

Then, we introduce $q(X)$ and $p(X)$ to facilitate the calculation of the gradient on $W$.

**Definition 25 (Definition of $q(X)$)** *If we have the below conditions,*

- *Let $h(X) \in \mathbb{R}^{n \times d}$ be defined as in Definition 23.*
- *Let $G_i \in \mathbb{R}^{n \times d}$ denote the gradient matrix resulting from the application of the chain rule up to the function $g_i$, i.e., $G_i = \frac{\mathrm{d}L(X)}{\mathrm{d}\mathsf{Attn}_i(T_{i-1}(X))}$.*
- *For $i_2 \in [n], j_2 \in [d]$, let $G_i(i_2, j_2)$ denote the $(i_2, j_2)$-th entry of $G_i$.*

*We define $q(X) \in \mathbb{R}^{n \times n}$ as*

$$q(X) = \underbrace{G_i}_{n \times d} \underbrace{h(X)^\top}_{d \times n}.$$

*where we define $q(X)_{j_0}^\top \in \mathbb{R}^n$ is the $j_0$-th row of $q(X)$.*

**Definition 26 (Definition of $p(X)$, Definition C.5 in Alman and Song [4])** *For every index $j_0 \in [n]$, we define $p(X)_{j_0} \in \mathbb{R}^n$ as*

$$p(X)_{j_0} := (\mathrm{diag}(f(X)_{j_0}) - f(X)_{j_0} f(X)_{j_0}^\top) q(X)_{j_0}$$

*where we have $p(X) \in \mathbb{R}^{n \times n}$ and we define $p(X)_{j_0}^\top \in \mathbb{R}^n$ is the $j_0$-th row of $p(X)$.*

*Furthermore, we define $p_1(X) = f(X) \odot q(X)$ and $p_2(X) = \mathrm{diag}(p_1(X) \cdot \mathbf{1}_n) f(X)$. Additionally, we can calculate $p(X)$ as*

$$p(X) = p_1(X) - p_2(X)$$

### F.4. Low rank representations

Using Alman and Song [3]'s polynomial method techniques, we can obtain the following low-rank representation result.

**Lemma 27 (Low rank representation to $f$, Section 3 of Alman and Song [3], Lemma D.1 of Alman and Song [4])** *For any $A = o(\sqrt{\log n})$, there exists a $k_1 = n^{o(1)}$ such that: Let $X \in \mathbb{R}^{n \times d}$ and $W \in \mathbb{R}^{d \times d}$ be a square matrix. It holds that $\|XW\|_\infty \leq R, \|X\|_\infty \leq R$, then there are two matrices $U_1, V_1 \in \mathbb{R}^{n \times k_1}$ such that $\|U_1 V_1^\top - f(X)\|_\infty \leq \epsilon / \mathrm{poly}(n)$. Here $f(X) = D^{-1} \exp(XWX^\top)$ (see also Definition 22) and we define $D = \mathrm{diag}(\exp(XWX^\top)\mathbf{1}_n)$ (see also Definition 21). Moreover, these matrices $U_1, V_1$ can be explicitly constructed in $n^{1+o(1)}$ time.*

A similar technique can be applied to $s(X)$.

**Lemma 28 (Low rank representation to $s$)** *Let $d = O(\log n)$. Assume that each number in the $n \times d$ matrices $h(X) \in \mathbb{R}^{n \times d}$ can be written using $O(\log n)$ bits. Let $n \times d$ matrix $s(X) \in \mathbb{R}^{n \times d}$ be defined as Definition 24. Then, there are two matrices $U_1, V_1 \in \mathbb{R}^{n \times k_1}$ we have $\|U_1 V_1^\top h(X) - s(X)\|_\infty \leq \epsilon / \mathrm{poly}(n)$.*

**Proof** We can show that

$$\begin{aligned}
\|U_1 V_1^\top h(X) - s(X)\|_\infty &= \|U_1 V_1^\top h(X) - f(X) h(X)\|_\infty \\
&= \|(\underbrace{U_1 V_1^\top}_{n \times n} - \underbrace{f(X)}_{n \times n}) \underbrace{h(X)}_{n \times d}\|_\infty \\
&\leq n \| \underbrace{U_1 V_1^\top}_{n \times n} - \underbrace{f(X)}_{n \times n} \|_\infty \| \underbrace{h(X)}_{n \times d} \|_\infty \\
&\leq n \| \underbrace{U_1 V_1^\top}_{n \times n} - \underbrace{f(X)}_{n \times n} \|_\infty \cdot \mathrm{poly}(n)
\end{aligned}$$

$$\leq \epsilon/\operatorname{poly}(n)$$

where the 1st step is from the choice of $s(X)$, the 2nd step comes from $AC - BC = (A - B)C$ holds for any matrices $A$, $B$, and $C$, the 3rd step is because of basic linear algebra, the 4th step is due to each number in $h(X)$ can be written using $O(\log(n))$ bits, the fifth step follows from $\|U_1 V_1^\top - f(X)\|_\infty \leq \epsilon/\operatorname{poly}(n)$. ∎

We can also get a low-rank representation of $p_1(x)$ and $p_2(x)$.

**Lemma 29 (Low rank representation to $p_1(X)$, Lemma D.4 of Alman and Song [4])** *Let $k_1 = n^{o(1)}$. Let $k_2 = n^{o(1)}$. Assume that $p_1(X) := f(X) \odot q(X)$. Assume $U_1, V_1 \in \mathbb{R}^{n \times k_1}$ approximates the $f(X)$ such that $\|U_1 V_1^\top - f(X)\|_\infty \leq \epsilon/\operatorname{poly}(n)$. Assume $U_2, V_2 \in \mathbb{R}^{n \times k_2}$ approximates the $q(X) \in \mathbb{R}^{n \times n}$ such that $\|U_2 V_2^\top - q(X)\|_\infty \leq \epsilon/\operatorname{poly}(n)$. Then there are matrices $U_3, V_3 \in \mathbb{R}^{n \times k_3}$ such that $\|U_3 V_3^\top - p_1(X)\|_\infty \leq \epsilon/\operatorname{poly}(n)$. The matrices $U_3, V_3$ can be explicitly constructed in $n^{1+o(1)}$ time.*

**Lemma 30 (Low rank representation $p_2(X)$, Lemma D.5 of Alman and Song [4])** *Let $k_1 = n^{o(1)}$. Let $k_2 = n^{o(1)}$. Let $k_4 = n^{o(1)}$. Assume that $p_2(X)$ is an $n \times n$ where $j_0$-th row $p_2(X)_{j_0} = f(X)_{j_0} f(X)_{j_0}^\top q(X)_{j_0}$ for each $j_0 \in [n]$. Assume $U_1, V_1 \in \mathbb{R}^{n \times k_1}$ approximates the $f(X)$ such that $\|U_1 V_1^\top - f(X)\|_\infty \leq \epsilon/\operatorname{poly}(n)$. Assume $U_2, V_2 \in \mathbb{R}^{n \times k_2}$ approximates the $q(X) \in \mathbb{R}^{n \times n}$ such that $\|U_2 V_2^\top - q(X)\|_\infty \leq \epsilon/\operatorname{poly}(n)$. Then there are matrices $U_4, V_4 \in \mathbb{R}^{n \times k_4}$ such that $\|U_4 V_4^\top - p_2(X)\|_\infty \leq \epsilon/\operatorname{poly}(n)$. The matrices $U_4, V_4$ can be explicitly constructed in $n^{1+o(1)}$ time.*

### F.5. Bounded entries of matrices

In this section, we provide proof that entries of matrices are bounded.

We begin with the exponential matrix $f(X)$.

**Lemma 31 (Bounded entries of $f(X)$)** *If we have the below conditions,*

- *Let $f(X) \in \mathbb{R}^{n \times n}$ be defined in Definition 22.*

*Then, we can show that*

$$\|f(X)\|_\infty \leq 1$$

**Proof** By Definition 22, we have

$$f(X) = \operatorname{diag}(\alpha(X))^{-1} u(X)$$

By Definition 21, we have

$$\alpha(X) = u(X)\mathbf{1}_n$$

Combining above two equations, we have

$$\|f(X)\|_\infty \leq 1$$

∎

A similar analysis can be applied to $h(X)$ and $s(X)$ as well.

**Lemma 32 (Bounded entries of $h(X)$)** *If we have the below conditions,*

- *Let $X \in \mathbb{R}^{n \times d}, W, W_V \in \mathbb{R}^{d \times d}$ be defined in Definition 2.*
- *Assuming each entry of $X, W, W_V$ can be re represented using $O(\log(n))$ bits.*
- *Let $h(X) \in \mathbb{R}^{n \times d}$ be defined in Definition 23.*

*Then, we can show that*

$$\|h(X)\|_\infty \leq \mathrm{poly}(n)$$

**Proof** By Definition 23, we have

$$h(X) := XW_V$$

Then, we have

$$\begin{aligned}
\|h(X)\|_\infty &= \|XW_V\|_\infty \\
&\leq n\|X\|_\infty \|W_V\|_\infty \\
&\leq \mathrm{poly}(n)
\end{aligned}$$

where the 1st step is from the definition of $h(X)$, the 2nd step comes from basic linear algebra, the 3rd step is because of each entry in $X$ and $W_V$ can be represented by $O(\log(n))$ bits. ■

**Lemma 33 (Bounded entries of $s(X)$)** *If we have the below conditions,*

- *Let $X \in \mathbb{R}^{n \times d}, W, W_V \in \mathbb{R}^{d \times d}$ be defined in Definition 2.*
- *Assuming each entry of $X, W, W_V$ can be re represented using $O(\log(n))$ bits.*
- *Let $s(X) \in \mathbb{R}^{n \times d}$ be defined in Definition 24.*

*Then, we can show that*

$$\|s(X)\|_\infty \leq \mathrm{poly}(n)$$

**Proof** By Definition 24, we have

$$\underbrace{s(X)}_{n \times d} = \underbrace{f(X)}_{n \times n} \underbrace{h(X)}_{n \times d}$$

Then, we have

$$\begin{aligned}
\|s(X)\|_\infty &= \|f(X)h(X)\|_\infty \\
&\leq n\|f(X)\|_\infty \|h(X)\|_\infty \\
&\leq \mathrm{poly}(n)
\end{aligned}$$

where the 1st step is from the definition of $c(X)$, the 2nd step comes from basic linear algebra, the 3rd step is because of Lemma 31, 32. ■

## Appendix G. Matrix View

In this section, we dive into analyzing the gradient of $\frac{\mathrm{d}L(X)}{\mathrm{d}T_{i-1}(X)}$.

In Section G.1, we give the gradient of $s(X)$ with respective to $X$. In Section G.2, we show the close form of the gradient on $T_i(X)$ via the chain rule. In Section G.3, we integrate each $C_i(X)$ to its corresponding matrix term $B_i(X)$. In Section G.4, applying the similar technique used in the previous section, we integrate the gradient on $T_i(X)$ into its corresponding matrix view. In Section G.5, we further apply matrix integration on each matrix term in the gradient on $T_i(X)$ calculated in the previous section. In Section G.6, we give the matrix view of all gradient components.

### G.1. Gradient of $s(X)$

In this section, we give the gradient of $s(X)$ with respective to $X$.

The results from Deng et al. [24] give the gradient of $c(X)$. By chain rule, the gradient of $s(X)$ is equivalent to the gradient of $c(X)$ from Deng et al. [24], since $c(X) = s(X) - B$ where $B$ is a constant matrix.

**Lemma 34 (Gradient of $s(X)_{i_0,j_0}$, Lemma B.16 in Deng et al. [24])** *If we have the below conditions,*

- *Let $s(X) \in \mathbb{R}^{n \times d}$ be defined as Definition 24*

*Then, we have*

- ***Part 1.*** *For all $i_0 = i_1 \in [n]$, $j_0, j_1 \in [d]$,*
$$\frac{\mathrm{d}s(X)_{i_0,j_0}}{\mathrm{d}X_{i_1,j_1}} = C_1(X) + C_2(X) + C_3(X) + C_4(X) + C_5(X)$$
  *where we have definitions:*
  - $C_1(X) := -s(X)_{i_0,j_0} \cdot f(X)_{i_0,i_0} \cdot \langle W_{j_1,*}, X_{i_0,*} \rangle$
  - $C_2(X) := -s(X)_{i_0,j_0} \cdot \langle f(X)_{i_0,*}, XW_{*,j_1} \rangle$
  - $C_3(X) := f(X)_{i_0,i_0} \cdot h(X)_{i_0,j_0} \cdot \langle W_{j_1,*}, X_{i_0,*} \rangle$
  - $C_4(X) := \langle f(X)_{i_0,*} \odot (XW_{*,j_1}), h(X)_{*,j_0} \rangle$
  - $C_5(X) := f(X)_{i_0,i_0} \cdot (W_V)_{j_1,j_0}$
- ***Part 2.*** *For all $i_0 \neq i_1 \in [n]$, $j_0, j_1 \in [d]$,*
$$\frac{\mathrm{d}s(X)_{i_0,j_0}}{\mathrm{d}X_{i_1,j_1}} = C_6(X) + C_7(X) + C_8(X)$$
  *where we have definitions:*
  - $C_6(X) := -s(X)_{i_0,j_0} \cdot f(X)_{i_1,i_0} \cdot \langle W_{j_1,*}, X_{i_0,*} \rangle$
    - *This is corresponding to $C_1(X)$*
  - $C_7(X) := f(X)_{i_1,i_0} \cdot h(X)_{i_1,j_0} \cdot \langle W_{j_1,*}, X_{i_0,*} \rangle$
    - *This is corresponding to $C_3(X)$*
  - $C_8(X) := f(X)_{i_1,i_0} \cdot (W_V)_{j_1,j_0}$
    - *This is corresponding to $C_5(X)$*

### G.2. Gradient on $T_i(X)$

In the Lemma 35, we use the chain rule to calculate the close form of the gradient on $T_i(X)$.

**Lemma 35 (Gradient for $T_i(X)$)** *If we have the below conditions,*

- *Let* $\mathsf{Attn}_i$ *be defined as Definition 2.*
- *Let* $T_i(X) \in \mathbb{R}^{n \times d}$ *be defined as Definition 6.*
- *Let* $s(X)$ *be defined as Definition 24.*
- *Let* $G_i \in \mathbb{R}^{n \times d}$ *denote the gradient matrix resulting from the application of the chain rule up to the function* $g_i$*, i.e.,* $G_i = \frac{\mathrm{d}L(X)}{\mathrm{d}\mathsf{Attn}_i(T_{i-1}(X))}$.
- *For* $i_2 \in [n], j_2 \in [d]$*, let* $G_i(i_2, j_2)$ *denote the* $(i_2, j_2)$*-th entry of* $G_i$.

*Then, we can show that, for* $i_1 \in [n]$*,* $j_1 \in [d]$*, we have*

$$\frac{\mathrm{d}L(X)}{\mathrm{d}T_{i-1}(X)_{i_1,j_1}} = \sum_{i_0=1}^{n} \sum_{j_0=1}^{d} G_i(i_0, j_0) \cdot \frac{\mathrm{d}s(X)_{i_0,j_0}}{\mathrm{d}X_{i_1,j_1}}$$

**Proof** By Lemma 18, we have

$$\frac{\mathrm{d}L(X)}{\mathrm{d}T_{i-1}(X)} = \sum_{i_2=1}^{n} \sum_{j_2=1}^{d} G_i(i_2, j_2) \cdot \frac{\mathrm{d}\mathsf{Attn}_i(T_{i-1}(X))_{i_2,j_2}}{\mathrm{d}T_{i-1}(X)}.$$

By Definition 2 and Definition 24, we have

$$\mathsf{Attn}_i(T_{i-1}(X)) = s(T_{i-1}(X))$$

Therefore, by combining above two equations and substituting variable $T_{i-1}(X) = X$, we have

$$\frac{\mathrm{d}L(X)}{\mathrm{d}T_{i-1}(X)_{i_1,j_1}} = \sum_{i_0=1}^{n} \sum_{j_0=1}^{d} G_i(i_0, j_0) \cdot \frac{\mathrm{d}s(X)_{i_0,j_0}}{\mathrm{d}X_{i_1,j_1}}$$

■

### G.3. Matrix view of $C(X)$

In this section, we will provide the matrix view of $C_i(X) \in \mathbb{R}$, for $i \in \{6, 7, 8, 2, 4\}$. We will consider each $C_i(X)$ one by one. We begin with $C_6(X)$.

**Lemma 36 (Matrix view of $C_6(X)$)** *If we have the below conditions,*

- *Let* $C_6(X, i_1, j_1) := -s(X)_{i_0,j_0} \cdot f(X)_{i_1,i_0} \cdot \langle W_{j_1,*}, X_{i_0,*} \rangle$ *be defined as in Lemma 34.*
- *We define a matrix* $B_6(X) \in \mathbb{R}^{n \times d}$*. For all* $i_1 \in [n], j_1 \in [d]$*, let* $B_6(i_1, j_1)$ *denote the* $(i_1, j_1)$*-th entry of* $B_6(X)$*. We define* $B_6(i_1, j_1) = C_6(X, i_1, j_1)$.

*Then, we can show that*

$$\underbrace{B_6(X)}_{n\times d} = \underbrace{-s(X)_{i_0,j_0}}_{1\times 1} \underbrace{f(X)_{*,i_0}}_{n\times 1} \underbrace{(W\cdot X_{i_0,*})^\top}_{1\times d}$$

**Proof** We have

$$\begin{aligned} C_6(X,i_1,j_1) = &~ -s(X)_{i_0,j_0}\cdot f(X)_{i_1,i_0}\cdot \langle W_{j_1,*}, X_{i_0,*}\rangle \\ = &~ -s(X)_{i_0,j_0}\cdot f(X)_{i_1,i_0}\cdot X_{i_0,*}^\top W_{j_1,*} \end{aligned}$$

where the 1st step is from the choice of $C_6(X)$, the 2nd step comes from $\langle a,b\rangle = a^\top b$ holds for any $a,b\in\mathbb{R}^d$.

We have

$$\underbrace{B_6(X)(i_1,*)}_{d\times 1} = ~ - \underbrace{s(X)_{i_0,j_0}}_{1\times 1} \underbrace{f(X)_{i_1,i_0}}_{1\times 1} \underbrace{W}_{d\times d} \underbrace{X_{i_0,*}}_{d\times 1}$$

Then, we have

$$\underbrace{B_6(X)}_{n\times d} = \underbrace{-s(X)_{i_0,j_0}}_{1\times 1} \underbrace{f(X)_{*,i_0}}_{n\times 1} \underbrace{(W\cdot X_{i_0,*})^\top}_{1\times d}$$

■

A similar analysis procedure can also be applied on $C_7(X)$.

**Lemma 37 (Matrix view of $C_7(X)$)** *If we have the below conditions,*

- *Let $C_7(X,i_1,j_1) := f(X)_{i_1,i_0}\cdot h(X)_{j_0,i_1}\cdot\langle W_{j_1,*}, X_{i_0,*}\rangle$ be defined as in Lemma 34.*
- *We define a matrix $B_7(X)\in\mathbb{R}^{n\times d}$. For all $i_1\in[n], j_1\in[d]$, let $B_7(i_1,j_1)$ denote the $(i_1,j_1)$-th entry of $B_7(X)$. We define $B_7(i_1,j_1) = C_7(X,i_1,j_1)$.*

*Then, we can show that*

$$\underbrace{B_7(X)}_{n\times d} = \underbrace{(f(X)_{*,i_0}\odot h(X)_{*,j_0})}_{n\times 1}\cdot \underbrace{(W\cdot X_{i_0,*})^\top}_{1\times d}$$

**Proof** We have

$$\begin{aligned} C_7(X,i_1,j_1) &= f(X)_{i_1,i_0}\cdot h(X)_{i_1,j_0}\cdot\langle W_{j_1,*}, X_{i_0,*}\rangle \\ &= f(X)_{i_1,i_0}\cdot h(X)_{i_1,j_0}\cdot W_{j_1,*}^\top X_{i_0,*} \end{aligned}$$

where the 1st step is from the choice of $C_7(X)$, the 2nd step comes from $\langle a,b\rangle = a^\top b$ holds for any $a,b\in\mathbb{R}^d$.

We have

$$B_7(X)(i_1,*) = f(X)_{i_1,i_0}\cdot h(X)_{i_1,j_0}\cdot W\cdot X_{i_0,*}$$

Then, we have

$$\underbrace{B_7(X)}_{n\times d} = \underbrace{(f(X)_{*,i_0} \odot h(X)_{*,j_0})}_{n\times 1} \cdot \underbrace{(W \cdot X_{i_0,*})^\top}_{1\times d}$$

■

Then, we provide an analysis of $C_8(X)$.

**Lemma 38 (Matrix view of $C_8(X)$)** *If we have the below conditions,*

- *Let $C_8(X, i_1, j_1) := f(X)_{i_1,i_0} \cdot (W_V)_{j_1,j_0}$ be defined as in Lemma 34.*
- *We define a matrix $B_8(X) \in \mathbb{R}^{n\times d}$. For all $i_1 \in [n], j_1 \in [d]$, let $B_8(i_1, j_1)$ denote the $(i_1, j_1)$-th entry of $B_8(X)$. We define $B_8(i_1, j_1) = C_8(X, i_1, j_1)$.*

*Then, we can show that*

$$\underbrace{B_8(X)}_{n\times d} = \underbrace{f(X)_{*,i_0}}_{n\times 1} \underbrace{(W_V)^\top_{*,j_0}}_{1\times d}$$

**Proof** We have

$$C_8(X, i_1, j_1) = f(X)_{i_1,i_0} \cdot (W_V)_{j_1,j_0}$$

where the 1st step is from the choice of $C_7(X)$.

We have

$$B_8(X)(i_1, *) = f(X)_{i_1,i_0} \cdot (W_V)_{*,j_0}$$

Then, we have

$$\underbrace{B_8(X)}_{n\times d} = \underbrace{f(X)_{*,i_0}}_{n\times 1} \underbrace{(W_V)^\top_{*,j_0}}_{1\times d}$$

■

Now, we consider $C_2(X)$.

**Lemma 39 (Matrix view of $C_2(X)$)** *If we have the below conditions,*

- *Let $C_2(X, j_1) := -s(X)_{i_0,j_0} \cdot \langle f(X)_{i_0,*}, XW_{*,j_1}\rangle$ be defined as in Lemma 34.*
- *We define a matrix $B_2(X) \in \mathbb{R}^d$. For all $j_1 \in [d]$, the $j_1$-th entry of $B_2(X)$ is defined as $C_2(X, j_1)$.*

*Then, we can show that*

$$\underbrace{B_2(X)}_{d\times 1} = \underbrace{-s(X)_{i_0,j_0}}_{1\times 1} \underbrace{W^\top}_{d\times d} \underbrace{X^\top}_{d\times n} \underbrace{f(X)_{i_0,*}}_{n\times 1}$$

**Proof** We have

$$\begin{aligned} C_2(X, j_1) = & - s(X)_{i_0,j_0} \cdot \langle f(X)_{i_0,*}, XW_{*,j_1} \rangle \\ = & - s(X)_{i_0,j_0} \cdot (XW_{*,j_1})^\top f(X)_{i_0,*} \\ = & \underbrace{-s(X)_{i_0,j_0}}_{1\times 1} \underbrace{W_{*,j_1}^\top}_{1\times d} \underbrace{X^\top}_{d\times n} \underbrace{f(X)_{i_0,*}}_{n\times 1} \end{aligned}$$

where the 1st step is from the choice of $C_2(X)$, the second step follows from $\langle a, b\rangle = a^\top b$, for any $a, b \in \mathbb{R}^n$.

Then, we have

$$\underbrace{B_2(X)}_{d\times 1} = \underbrace{-s(X)_{i_0,j_0}}_{1\times 1} \underbrace{W^\top}_{d\times d} \underbrace{X^\top}_{d\times n} \underbrace{f(X)_{i_0,*}}_{n\times 1}$$

■

Finally, we analyze $C_4(X)$, which is the last term we need to compute.

**Lemma 40 (Matrix view of $C_4(X)$)** *If we have the below conditions,*

- *Let $C_4(X, j_1) := \langle f(X)_{i_0,*} \odot (XW_{*,j_1}), h(X)_{*,j_0} \rangle$ be defined as in Lemma 34.*
- *We define a matrix $B_4(X) \in \mathbb{R}^d$. For all $j_1 \in [d]$, the $j_1$-th entry of $B_4(X)$ is defined as $C_4(X, j_1)$.*

*Then, we can show that*

$$\underbrace{B_4(X)}_{d\times 1} = \underbrace{W^\top}_{d\times d} \underbrace{X^\top}_{d\times n} \underbrace{(f(X)_{i_0,*} \odot h(X)_{*,j_0})}_{n\times 1}$$

**Proof** We have

$$\begin{aligned} C_4(X, j_1) &= \langle f(X)_{i_0,*} \odot (XW_{*,j_1}), h(X)_{*,j_0} \rangle \\ &= \langle f(X)_{i_0,*} \odot h(X)_{*,j_0}, (XW_{*,j_1}) \rangle \\ &= (XW_{*,j_1})^\top (f(X)_{i_0,*} \odot h(X)_{*,j_0}) \end{aligned}$$

where the 1st step is from the choice of $C_4(X)$, the 2nd step comes from Fact 16, and the last step follows from basic linear algebra. ■

### G.4. Matrix view of gradient on $T_i(X)$

Since we have got the matrix view of each $C_i(X)$ term in the previous section, we can get the matrix view of the gradient on $T_i(X)$ in Lemma 41.

**Lemma 41 (Matrix view of single entry of gradient)** *If we have the below conditions,*

- *Let $s(X)$ be defined as Definition 24.*

- *Let $G_i \in \mathbb{R}^{n \times d}$ denote the gradient matrix resulting from the application of the chain rule up to the function $g_i$, i.e., $G_i = \frac{\mathrm{d}L(X)}{\mathrm{dAttn}_i(T_{i-1}(X))}$.*
- *For $i_2 \in [n], j_2 \in [d]$, let $G_i(i_2, j_2)$ denote the $(i_2, j_2)$-th entry of $G_i$.*
- *Let $B_6(X), B_7(X), B_8(X) \in \mathbb{R}^{n \times d}$ be defined in Lemma 36, Lemma 37, and Lemma 38*
- *Let $B_2(X), B_4(X) \in \mathbb{R}^d$ be defined in Lemma 39 and Lemma 40.*

*For any $i_0 \in [n], j_0 \in [d]$, we have*

$$G_i(i_0, j_0) \cdot \frac{\mathrm{d}s(X)_{i_0,j_0}}{\mathrm{d}X} = \underbrace{G_i(i_0, j_0)}_{1 \times 1} \cdot (\underbrace{B_6(X) + B_7(X) + B_8(X)}_{n \times d} + \underbrace{e_{i_0}}_{n \times 1} \underbrace{(B_2(X) + B_4(X))^\top}_{1 \times d})$$

**Proof**

By Lemma 34, we have

- **Part 1.** For all $i_0 = i_1 \in [n]$, $j_0, j_1 \in [d]$,

$$\frac{\mathrm{d}s(X)_{i_0,j_0}}{\mathrm{d}X_{i_1,j_1}} = C_1(X) + C_2(X) + C_3(X) + C_4(X) + C_5(X) \tag{1}$$

- **Part 2.** For all $i_0 \neq i_1 \in [n]$, $j_0, j_1 \in [d]$,

$$\frac{\mathrm{d}s(X)_{i_0,j_0}}{\mathrm{d}X_{i_1,j_1}} = C_6(X) + C_7(X) + C_8(X) \tag{2}$$

Since for any $i_1 \in [n]$, $j_1 \in [d]$, let $G_i(i_0, j_0) \cdot \frac{\mathrm{d}s(X)_{i_0,j_0}}{\mathrm{d}X_{i_1,j_1}}$ denote the $(i_1, j_1)$-th entry of $G_i(i_0, j_0) \cdot \frac{\mathrm{d}s(X)_{i_0,j_0}}{\mathrm{d}X}$, we consider the following two cases:

- **Case 1.** The $i_0$-th row of $G_i(i_0, j_0) \cdot \frac{\mathrm{d}s(X)_{i_0,j_0}}{\mathrm{d}X}$.
- **Case 2.** The other $n - 1$ rows of $G_i(i_0, j_0) \cdot \frac{\mathrm{d}s(X)_{i_0,j_0}}{\mathrm{d}X}$ where $i_1 \neq i_0$.

We first consider **Case 1.**

Recall that the matrix view of $C_2(X), C_4(X) \in \mathbb{R}$ are $B_2(X), B_4(X) \in \mathbb{R}^d$, and the matrix view of $C_6(X), C_7(X), C_8(X) \in \mathbb{R}$ are $B_6(X), B_7(X), B_8(X) \in \mathbb{R}^{n \times d}$, respectively.

For $k \in \{6, 7, 8\}$, we use $B_k(X)(s, *) \in \mathbb{R}^d$ to denote the $s$-th row of $B_k(X)$.

We use $(G_i(i_0, j_0) \cdot \frac{\mathrm{d}s(X)_{i_0,j_0}}{\mathrm{d}X})(i_0, *) \in \mathbb{R}^d$ to denote the $i_0$-th row of $G_i(i_0, j_0) \cdot \frac{\mathrm{d}s(X)_{i_0,j_0}}{\mathrm{d}X}$.

Since $C_6(X), C_7(X), C_8(X)$ are the corresponding parts of $C_1(X), C_3(X), C_5(X)$, and by Eq. (1), then we can have the following

$$(G_i(i_0, j_0) \cdot \frac{\mathrm{d}s(X)_{i_0,j_0}}{\mathrm{d}X})(i_0, *) = \underbrace{G_i(i_0, j_0)}_{1 \times 1} \cdot \underbrace{(B_6(X)(i_0, *) + B_7(X)(i_0, *) + B_8(X)(i_0, *) + B_2(X) + B_4(X))}_{d \times 1}$$

We then consider **Case 2.**

For $k \in \{6, 7, 8\}$, we use $B_k(X)(\neq s, *) \in \mathbb{R}^{(n-1)\times d}$ to denote the matrix $B_k(X)$ with the $s$-th row removed.

Similarly, we use $(G_i(i_0, j_0) \cdot \frac{\mathrm{d}s(X)_{i_0,j_0}}{\mathrm{d}X})(\neq i_0, *) \in \mathbb{R}^{(n-1)\times d}$ to denote the matrix $G_i(i_0, j_0) \cdot \frac{\mathrm{d}s(X)_{i_0,j_0}}{\mathrm{d}X}$ with the $i_0$-th row removed.

By Eq. (2), we have

$$(G_i(i_0, j_0) \cdot \frac{\mathrm{d}s(X)_{i_0,j_0}}{\mathrm{d}X})(\neq i_0, *) = \underbrace{G_i(i_0, j_0)}_{1\times 1} \cdot \underbrace{(B_6(X)(\neq i_0, *) + B_7(X)(\neq i_0, *) + B_8(X)(\neq i_0, *))}_{d\times(n-1)}$$

Combining **Case 1.** and **Case 2.** together, we have

$$G_i(i_0, j_0) \cdot \frac{\mathrm{d}s(X)_{i_0,j_0}}{\mathrm{d}X} = \underbrace{G_i(i_0, j_0)}_{1\times 1} \cdot (\underbrace{B_6(X) + B_7(X) + B_8(X)}_{n\times d} + \underbrace{e_{i_0}}_{n\times 1} \underbrace{(B_2(X) + B_4(X))^\top}_{1\times d})$$

■

Then, we have the matrix view of $T_i(X)$ gradient.

**Lemma 42 (Matrix view of $T_i(X)$ gradient)** *If we have the below conditions,*

- *Let $L(X)$ be defined as Definition 5.*
- *Let $T(X)$ be defined as Definition 6.*
- *Let $G_i \in \mathbb{R}^{n\times d}$ denote the gradient matrix resulting from the application of the chain rule up to the function $g_i$, i.e., $G_i = \frac{\mathrm{d}L(X)}{\mathrm{d}\mathsf{Attn}_i(T_{i-1}(X))}$.*
- *For $i_2 \in [n], j_2 \in [d]$, let $G_i(i_2, j_2)$ denote the $(i_2, j_2)$-th entry of $G_i$.*
- *Let $B_6(X), B_7(X), B_8(X) \in \mathbb{R}^{n\times d}$ be defined in Lemma 36, Lemma 37, and Lemma 38*
- *Let $B_2(X), B_4(X) \in \mathbb{R}^d$ be defined in Lemma 39 and Lemma 40.*

*Then, we have*

$$\frac{\mathrm{d}L(X)}{\mathrm{d}T_{i-1}(X)} = \sum_{i_0=1}^{n}\sum_{j_0=1}^{d} \underbrace{G_i(i_0, j_0)}_{1\times 1} \cdot (\underbrace{B_6(X) + B_7(X) + B_8(X)}_{n\times d} + \underbrace{e_{i_0}}_{n\times 1} \underbrace{(B_2(X) + B_4(X))^\top}_{1\times d})$$

**Proof** By Lemma 41, we have

$$G_i(i_0, j_0) \cdot \frac{\mathrm{d}s(X)_{i_0,j_0}}{\mathrm{d}X} = \underbrace{G_i(i_0, j_0)}_{1\times 1} \cdot (\underbrace{B_6(X) + B_7(X) + B_8(X)}_{n\times d} + \underbrace{e_{i_0}}_{n\times 1} \underbrace{(B_2(X) + B_4(X))^\top}_{1\times d})$$

Then, by Lemma 18 we have

$$\frac{\mathrm{d}L(X)}{\mathrm{d}T_{i-1}(X)} = \sum_{i_2=1}^{n}\sum_{j_2=1}^{d} G_i(i_2, j_2) \cdot \frac{\mathrm{d}\mathsf{Attn}_i(T_{i-1}(X))_{i_2,j_2}}{\mathrm{d}T_{i-1}(X)}.$$

After combining the above two equations, we are done. ■

### G.5. Matrix view of each term in gradient on $T_i(X)$

In this subsection, we reduce the double summation to a matrix product for easy and clear analysis.

We first work on the $B_6$ term.

**Lemma 43 (Matrix view of $B_6(X)$ term)** *If we have the below conditions,*

- *Let* $\underbrace{B_6(X)}_{n\times d} = \underbrace{-s(X)_{i_0,j_0}}_{1\times 1}\underbrace{f(X)_{*,i_0}}_{n\times 1}\underbrace{(W\cdot X_{i_0,*})^\top}_{1\times d}$ *be defined in Lemma 36.*
- *We define $z_6(X)\in\mathbb{R}^{n\times n}$, which satisfies*

$$\underbrace{z_6(X)_{*,i_0}}_{n\times 1} = (\underbrace{G_i(i_0,*)^\top}_{1\times d}\underbrace{s(X)_{i_0,*}}_{d\times 1})\underbrace{f(X)_{*,i_0}}_{n\times 1}$$

- *Let $f(X)\in\mathbb{R}^{n\times n}$ be defined in Definition 22.*
- *Let $W\in\mathbb{R}^{d\times d}$ be defined in Definition 2.*
- *Let $G_i\in\mathbb{R}^{n\times d}$ denote the gradient matrix resulting from the application of the chain rule up to the function $g_i$, i.e., $G_i = \frac{\mathrm{d}L(X)}{\mathrm{d}\mathsf{Attn}_i(T_{i-1}(X))}$.*
- *For $i_2\in[n], j_2\in[d]$, let $G_i(i_2,j_2)$ denote the $(i_2,j_2)$-th entry of $G_i$.*

*Then we have*

$$\sum_{i_0=1}^n\sum_{j_0=1}^d \underbrace{G_i(i_0,j_0)}_{1\times 1}\underbrace{B_6(X)}_{n\times d} = -\underbrace{z_6(X)}_{n\times n}\underbrace{X}_{n\times d}\underbrace{W^\top}_{d\times d}$$

**Proof**

$$\begin{aligned}
\sum_{i_0=1}^n\sum_{j_0=1}^d G_i(i_0,j_0)B_6(X) = &\ -\sum_{i_0=1}^n\sum_{j_0=1}^d \underbrace{G_i(i_0,j_0)}_{1\times 1}\underbrace{s(X)_{i_0,j_0}}_{1\times 1}\underbrace{f(X)_{*,i_0}}_{n\times 1}\underbrace{(W\cdot X_{i_0,*})^\top}_{1\times d}\\
= &\ -\sum_{i_0=1}^n(\sum_{j_0=1}^d \underbrace{G_i(i_0,j_0)}_{1\times 1}\underbrace{s(X)_{i_0,j_0}}_{1\times 1})\underbrace{f(X)_{*,i_0}}_{n\times 1}\underbrace{(W\cdot X_{i_0,*})^\top}_{1\times d}\\
= &\ -\sum_{i_0=1}^n(\underbrace{G_i(i_0,*)^\top}_{1\times d}\underbrace{s(X)_{i_0,*}}_{d\times 1})\underbrace{f(X)_{*,i_0}}_{n\times 1}\underbrace{(W\cdot X_{i_0,*})^\top}_{1\times d}\\
= &\ -\sum_{i_0=1}^n(\underbrace{G_i(i_0,*)^\top}_{1\times d}\underbrace{s(X)_{i_0,*}}_{d\times 1})\underbrace{f(X)_{*,i_0}}_{n\times 1}\underbrace{X_{i_0,*}^\top}_{1\times d}\underbrace{W^\top}_{d\times d}
\end{aligned}$$

where the 1st step is from the choice of $B_6(X)$, the 2nd step comes from basic algebra, the 3rd step is because of $a^\top b = \sum_{i=1}^d a_i\cdot b_i$ holds for any $a,b\in\mathbb{R}^d$, the 4th step is due to $(AB)^\top = B^\top A^\top$ for any matrices $A$ and $B$.

Recall that we have $\underbrace{z_6(X)_{*,i_0}}_{n\times 1} = (\underbrace{G_i(i_0,*)^\top}_{1\times d} \underbrace{s(X)_{i_0,*}}_{d\times 1}) \underbrace{f(X)_{*,i_0}}_{n\times 1}$.

Then, we have

$$\begin{aligned}
& -\sum_{i_0=1}^n (\underbrace{G_i(i_0,*)^\top}_{1\times d} \underbrace{s(X)_{i_0,*}}_{d\times 1}) \underbrace{f(X)_{*,i_0}}_{n\times 1} \underbrace{X_{i_0,*}^\top}_{1\times d} \underbrace{W^\top}_{d\times d} \\
= & -\sum_{i_0=1}^n \underbrace{z_6(X)_{*,i_0}}_{n\times 1} \underbrace{X_{i_0,*}^\top}_{1\times d} \underbrace{W^\top}_{d\times d} \\
= & -\underbrace{z_6(X)}_{n\times n} \underbrace{X}_{n\times d} \underbrace{W^\top}_{d\times d}
\end{aligned}$$

where the 1st step is from the choice of $z_6(X)$, the 2nd step comes from basic linear algebra. ■

Then, we can get the matrix view of $B_7(X)$ term.

**Lemma 44 (Matrix view of $B_7(X)$ term)** *If we have the below conditions,*

- *Let* $\underbrace{B_7(X)}_{n\times d} = \underbrace{(f(X)_{*,i_0} \odot h(X)_{*,j_0})}_{n\times 1} \cdot \underbrace{(W \cdot X_{i_0,*})^\top}_{1\times d}$ *be defined in Lemma 37.*
- *We define* $z_7(X) \in \mathbb{R}^{n\times n}$*, which satisfies*

$$\underbrace{z_7(X)_{*,i_0}}_{n\times 1} = \underbrace{f(X)_{*,i_0}}_{n\times 1} \odot (\underbrace{h(X)}_{n\times d} \underbrace{G_i(i_0,*)}_{d\times 1}).$$

- *Let* $X \in \mathbb{R}^{n\times d}, W \in \mathbb{R}^{d\times d}$ *be defined in Definition 2.*
- *Let* $G_i \in \mathbb{R}^{n\times d}$ *denote the gradient matrix resulting from the application of the chain rule up to the function* $g_i$*, i.e.,* $G_i = \frac{\mathrm{d}L(X)}{\mathrm{dAttn}_i(T_{i-1}(X))}$.
- *For* $i_2 \in [n], j_2 \in [d]$*, let* $G_i(i_2, j_2)$ *denote the* $(i_2, j_2)$*-th entry of* $G_i$*.*

*Then we have*

$$\sum_{i_0=1}^n \sum_{j_0=1}^d \underbrace{G_i(i_0,j_0)}_{1\times 1} \underbrace{B_7(X)}_{n\times d} = \underbrace{z_7(X)}_{n\times n} \underbrace{X}_{n\times d} \underbrace{W^\top}_{d\times d}$$

**Proof** We have

$$\begin{aligned}
\sum_{i_0=1}^n \sum_{j_0=1}^d \underbrace{G_i(i_0,j_0)}_{1\times 1} \underbrace{B_7(X)}_{n\times d} = & \sum_{i_0=1}^n \sum_{j_0=1}^d \underbrace{G_i(i_0,j_0)}_{1\times 1} \underbrace{(f(X)_{*,i_0} \odot h(X)_{*,j_0})}_{n\times 1} \cdot \underbrace{(W \cdot X_{i_0,*})^\top}_{1\times d} \\
= & \sum_{i_0=1}^n (\underbrace{f(X)_{*,i_0}}_{n\times 1} \odot (\sum_{j_0=1}^d \underbrace{G_i(i_0,j_0)}_{1\times 1} \underbrace{h(X)_{*,j_0}}_{n\times 1})) \cdot \underbrace{(W \cdot X_{i_0,*})^\top}_{1\times d}
\end{aligned}$$

$$= \sum_{i_0=1}^n (\underbrace{f(X)_{*,i_0}}_{n\times 1} \odot (\underbrace{h(X)}_{n\times d} \underbrace{G_i(i_0,*)}_{d\times 1})) \cdot \underbrace{(X_{i_0,*}^\top W^\top)}_{1\times d}$$

where the 1st step is from the choice of $B_7(X)$, the 2nd step comes from basic algebra, the 3rd step is because of basic linear algebra.

Recall that we have $\underbrace{z_7(X)_{*,i_0}}_{n\times 1} = \underbrace{f(X)_{*,i_0}}_{n\times 1} \odot (\underbrace{h(X)}_{n\times d} \underbrace{G_i(i_0,*)}_{d\times 1})$.

Then we have

$$\begin{aligned}
&\sum_{i_0=1}^n (\underbrace{f(X)_{*,i_0}}_{n\times 1} \odot (\underbrace{h(X)}_{n\times d} \underbrace{G_i(i_0,*)}_{d\times 1})) \cdot \underbrace{(X_{i_0,*}^\top W^\top)}_{1\times d} \\
= & \sum_{i_0=1}^n \underbrace{z_7(X)_{*,i_0}}_{n\times 1} \underbrace{X_{i_0,*}^\top}_{1\times d} \underbrace{W^\top}_{d\times d} \\
= & \underbrace{z_7(X)}_{n\times n} \underbrace{X}_{n\times d} \underbrace{W^\top}_{d\times d}
\end{aligned}$$

where the 1st step is from the choice of $z_7(X)$, the 2nd step comes from basic linear algebra. ■

Then, we consider $B_8(X)$.

**Lemma 45 (Matrix view of $B_8(X)$ term)** *If we have the below conditions,*

- *Let* $\underbrace{B_8(X)}_{n\times d} = \underbrace{f(X)_{*,i_0}}_{n\times 1} \underbrace{(W_V)_{*,j_0}^\top}_{1\times d}$ *be defined in Lemma 38.*
- *Let* $G_i \in \mathbb{R}^{n\times d}$ *denote the gradient matrix resulting from the application of the chain rule up to the function* $g_i$*, i.e.,* $G_i = \frac{\mathrm{d}L(X)}{\mathrm{dAttn}_i(T_{i-1}(X))}$.
- *For* $i_2 \in [n], j_2 \in [d]$*, let* $G_i(i_2, j_2)$ *denote the* $(i_2, j_2)$*-th entry of* $G_i$.

*Then we have*

$$\sum_{i_0=1}^n \sum_{j_0=1}^d \underbrace{G_i(i_0,j_0)}_{1\times 1} \underbrace{B_8(X)}_{n\times d} = \underbrace{f(X)}_{n\times n} \underbrace{G_i}_{n\times d} \underbrace{W_V^\top}_{d\times d}$$

**Proof** We have

$$\begin{aligned}
\sum_{i_0=1}^n \sum_{j_0=1}^d \underbrace{G_i(i_0,j_0)}_{1\times 1} \underbrace{B_8(X)}_{n\times d} = & \sum_{i_0=1}^n \sum_{j_0=1}^d \underbrace{G_i(i_0,j_0)}_{1\times 1} \underbrace{f(X)_{*,i_0}}_{n\times 1} \underbrace{(W_V)_{*,j_0}^\top}_{1\times d} \\
= & \sum_{i_0=1}^n \underbrace{f(X)_{*,i_0}}_{n\times 1} (\sum_{j_0=1}^d \underbrace{G_i(i_0,j_0)}_{1\times 1} \underbrace{(W_V)_{*,j_0}^\top}_{1\times d})
\end{aligned}$$

$$= \sum_{i_0=1}^{n} \underbrace{f(X)_{*,i_0}}_{n\times 1} \underbrace{G_i(i_0,*)^\top}_{1\times d} \underbrace{W_V^\top}_{d\times d}$$
$$= \underbrace{f(X)}_{n\times n} \underbrace{G_i}_{n\times d} \underbrace{W_V^\top}_{d\times d}$$

where the 1st step is from the choice of $B_8(X)$, the 2nd step comes from basic algebra, the 3rd step is because of basic linear algebra, the 4th step is due to basic linear algebra.

■

Now, we can do the matrix view of $B_2(X)$ term.

**Lemma 46 (Matrix view of $B_2(X)$ term)** *If we have the below conditions,*

- *Let* $\underbrace{B_2(X)}_{d\times 1} = \underbrace{-s(X)_{i_0,j_0}}_{1\times 1} \underbrace{W^\top}_{d\times d} \underbrace{X^\top}_{d\times n} \underbrace{f(X)_{i_0,*}}_{n\times 1}$ *be defined in Lemma 39*
- *Let $G_i \in \mathbb{R}^{n\times d}$ denote the gradient matrix resulting from the application of the chain rule up to the function $g_i$, i.e., $G_i = \frac{\mathrm{d}L(X)}{\mathrm{dAttn}_i(T_{i-1}(X))}$.*
- *For $i_2 \in [n], j_2 \in [d]$, let $G_i(i_2, j_2)$ denote the $(i_2, j_2)$-th entry of $G_i$.*
- *We define $z_2(X) \in \mathbb{R}^{n\times n}$, which satisfies*

$$\underbrace{z_2(X)_{i_0,*}}_{n\times 1} = (\underbrace{G_i(i_0,*)^\top}_{1\times d} \underbrace{s(X)_{i_0,*}}_{d\times 1}) \underbrace{f(X)_{i_0,*}}_{n\times 1}$$

- *Let $X \in \mathbb{R}^{n\times d}, W \in \mathbb{R}^{d\times d}$ be defined in Definition 2*

*Then we have*

$$\sum_{i_0=1}^{n}\sum_{j_0=1}^{d} \underbrace{G_i(i_0,j_0)}_{1\times 1} \underbrace{e_{i_0}}_{n\times 1} \underbrace{B_2(X)^\top}_{1\times d} = -\underbrace{z_2(X)}_{n\times n} \underbrace{X}_{n\times d} \underbrace{W}_{d\times d}$$

**Proof** We have

$$\sum_{i_0=1}^{n}\sum_{j_0=1}^{d} \underbrace{G_i(i_0,j_0)}_{1\times 1} \underbrace{e_{i_0}}_{n\times 1} \underbrace{B_2(X)^\top}_{1\times d} = -\sum_{i_0=1}^{n}\sum_{j_0=1}^{d} \underbrace{G_i(i_0,j_0)}_{1\times 1} \underbrace{s(X)_{i_0,j_0}}_{1\times 1} \underbrace{e_{i_0}}_{n\times 1} \underbrace{f(X)_{i_0,*}^\top}_{1\times n} \underbrace{X}_{n\times d} \underbrace{W}_{d\times d}$$
$$= -\sum_{i_0=1}^{n} (\sum_{j_0=1}^{d} \underbrace{G_i(i_0,j_0)}_{1\times 1} \underbrace{s(X)_{i_0,j_0}}_{1\times 1}) \underbrace{e_{i_0}}_{n\times 1} \underbrace{f(X)_{i_0,*}^\top}_{1\times n} \underbrace{X}_{n\times d} \underbrace{W}_{d\times d}$$
$$= -\sum_{i_0=1}^{n} (\underbrace{G_i(i_0,*)^\top}_{1\times d} \underbrace{s(X)_{i_0,*}}_{d\times 1}) \underbrace{e_{i_0}}_{n\times 1} \underbrace{f(X)_{i_0,*}^\top}_{1\times n} \underbrace{X}_{n\times d} \underbrace{W}_{d\times d}$$
$$= -\sum_{i_0=1}^{n} \underbrace{e_{i_0}}_{n\times 1} (\underbrace{G_i(i_0,*)^\top}_{1\times d} \underbrace{s(X)_{i_0,*}}_{d\times 1}) \underbrace{f(X)_{i_0,*}^\top}_{1\times n} \underbrace{X}_{n\times d} \underbrace{W}_{d\times d}$$

where the 1st step is from the choice of $B_2(X)$, the 2nd step comes from basic algebra, the 3rd step is because of $a^\top b = \sum_{i=1}^d a_i \cdot b_i$ holds for any $a, b \in \mathbb{R}^d$, the 4th step is due to $(AB)^\top = B^\top A^\top$ holds for any matrix $A, B$.

Recall that we have $\underbrace{z_2(X)_{i_0,*}}_{n\times 1} = (\underbrace{G_i(i_0,*)^\top}_{1\times d} \underbrace{s(X)_{i_0,*}}_{d\times 1}) \underbrace{f(X)_{i_0,*}}_{n\times 1}$.

Then, we have

$$\begin{aligned}
&- \sum_{i_0=1}^n \underbrace{e_{i_0}}_{n\times 1} (\underbrace{G_i(i_0,*)^\top}_{1\times d} \underbrace{s(X)_{i_0,*}}_{d\times 1}) \underbrace{f(X)_{i_0,*}^\top}_{1\times n} \underbrace{X}_{n\times d} \underbrace{W}_{d\times d} \\
=& - \sum_{i_0=1}^n \underbrace{e_{i_0}}_{n\times 1} \underbrace{z_2(X)_{i_0,*}^\top}_{1\times n} \underbrace{X}_{n\times d} \underbrace{W}_{d\times d} \\
=& - \underbrace{z_2(X)}_{n\times n} \underbrace{X}_{n\times d} \underbrace{W}_{d\times d}
\end{aligned}$$

where the 1st step is from the choice of $z_2(X)$, the 2nd step comes from basic linear algebra. ■

Finally, we do a similar analysis for the term $B_4(X)$. Then, we get all the matrix views we need.

**Lemma 47 (Matrix view of $B_4(X)$ term)** *If we have the below conditions,*

- *Let $\underbrace{B_4(X)}_{d\times 1} = \underbrace{W^\top}_{d\times d} \underbrace{X^\top}_{d\times n} \underbrace{(f(X)_{i_0,*} \odot h(X)_{*,j_0})}_{n\times 1}$ be defined in Lemma 40.*
- *Let $G_i \in \mathbb{R}^{n\times d}$ denote the gradient matrix resulting from the application of the chain rule up to the function $g_i$, i.e., $G_i = \frac{\mathrm{d}L(X)}{\mathrm{d}\mathsf{Attn}_i(T_{i-1}(X))}$.*
- *For $i_2 \in [n], j_2 \in [d]$, let $G_i(i_2, j_2)$ denote the $(i_2, j_2)$-th entry of $G_i$.*
- *We define $z_4(X) \in \mathbb{R}^{n\times n}$, which satisfies*

$$\underbrace{z_4(X)_{i_0,*}}_{n\times 1} = \underbrace{f(X)_{i_0,*}}_{n\times 1} \odot \underbrace{(h(X)G_i(i_0,*))}_{n\times 1}$$

*Then we have*

$$\sum_{i_0=1}^n \sum_{j_0=1}^d \underbrace{G_i(i_0,j_0)}_{1\times 1} \underbrace{e_{i_0}}_{n\times 1} \underbrace{B_4(X)^\top}_{1\times d} = \underbrace{z_4(X)}_{n\times n} \underbrace{X}_{n\times d} \underbrace{W}_{d\times d}$$

**Proof** We have

$$\begin{aligned}
&\sum_{i_0=1}^n \sum_{j_0=1}^d \underbrace{G_i(i_0,j_0)}_{1\times 1} \underbrace{e_{i_0}}_{n\times 1} \underbrace{B_4(X)^\top}_{1\times d} \\
=& \sum_{i_0=1}^n \sum_{j_0=1}^d \underbrace{G_i(i_0,j_0)}_{1\times 1} \underbrace{e_{i_0}}_{n\times 1} \underbrace{(f(X)_{i_0,*}^\top \odot h(X)_{*,j_0}^\top)}_{1\times n} \underbrace{X}_{n\times d} \underbrace{W}_{d\times d}
\end{aligned}$$

$$= \sum_{i_0=1}^{n} \underbrace{e_{i_0}}_{n\times 1} (\underbrace{f(X)_{i_0,*}^\top}_{1\times n} \odot (\sum_{j_0=1}^{d} \underbrace{G_i(i_0,j_0)}_{1\times 1} \underbrace{h(X)_{*,j_0}^\top}_{1\times n})) \underbrace{X}_{n\times d} \underbrace{W}_{d\times d}$$

$$= \sum_{i_0=1}^{n} \underbrace{e_{i_0}}_{n\times 1} (\underbrace{f(X)_{i_0,*}^\top}_{1\times n} \odot \underbrace{(h(X)G_i(i_0,*))^\top}_{1\times n}) \underbrace{X}_{n\times d} \underbrace{W}_{d\times d}$$

$$= \sum_{i_0=1}^{n} \underbrace{e_{i_0}}_{n\times 1} \underbrace{z_4(X)_{i_0,*}^\top}_{1\times n} \underbrace{X}_{n\times d} \underbrace{W}_{d\times d}$$

$$= \underbrace{z_4(X)}_{n\times n} \underbrace{X}_{n\times d} \underbrace{W}_{d\times d}$$

where the 1st step is from the choice of $B_4(X)$, the 2nd step comes from basic algebra, the 3rd step is because of basic linear algebra, the 4th step is due to the choice of $z_4(X)$, the 5th step follows from basic linear algebra. ■

### G.6. Components of gradient on $T_i(X)$

**Definition 48 (Definition of $D_k$)** *If we have the below conditions,*

- *For $k_1 \in \{6,7,8\}$, let $B_{k_1}(X) \in \mathbb{R}^{n\times d}$ be defined as Lemma 36, 37, and 38, respectively.*
- *For $k_2 \in \{2,4\}$, let $B_{k_2}(X) \in \mathbb{R}^{d\times 1}$ be defined as Lemma 39 and 40, respectively.*
- *Let $G_i \in \mathbb{R}^{n\times d}$ denote the gradient matrix resulting from the application of the chain rule up to the function $g_i$, i.e., $G_i = \frac{\mathrm{d}L(X)}{\mathrm{d}\mathsf{Attn}_i(T_{i-1}(X))}$.*

*We define $D_k \in \mathbb{R}^{n\times d}$ as follows:*

- *For $k_1 \in \{6,7,8\}$, we define*

$$D_{k_1} := \sum_{i_0=1}^{n} \sum_{j_0=1}^{d} \underbrace{G_i(i_0,j_0)}_{1\times 1} \underbrace{B_{k_1}(X)}_{n\times d}$$

- *For $k_2 \in \{2,4\}$, we define*

$$D_{k_2} := \sum_{i_0=1}^{n} \sum_{j_0=1}^{d} \underbrace{G_i(i_0,j_0)}_{1\times 1} \underbrace{e_{i_0}}_{n\times 1} \underbrace{B_{k_2}(X)^\top}_{1\times d}$$

**Definition 49 (Definition of $K$)** *If we have the below conditions,*

- *Let $s(X) \in \mathbb{R}^{n\times d}$ be defined as Definition 24.*
- *Let $G_i \in \mathbb{R}^{n\times d}$ denote the gradient matrix resulting from the application of the chain rule up to the function $g_i$, i.e., $G_i = \frac{\mathrm{d}L(X)}{\mathrm{d}\mathsf{Attn}_i(T_{i-1}(X))}$.*

*We define $K \in \mathbb{R}^n$, where for each $i_0 \in [n]$, we define*

$$\underbrace{K_{i_0}}_{1 \times 1} = \underbrace{G_i(i_0, *)^\top}_{1 \times d} \underbrace{s(X)_{i_0,*}}_{d \times 1}$$

*Furthermore, we have*

$$\underbrace{K}_{n \times 1} = \underbrace{(G_i \odot s(X))}_{n \times d} \underbrace{\mathbf{1}_d}_{d \times 1}$$

**Lemma 50 (Close form of $D_k$)** *If we have the below conditions,*

- *Let $X \in \mathbb{R}^{n \times d}, W \in \mathbb{R}^{d \times d}$ be defined as Definition 2.*
- *For $k \in \{6, 7, 8, 2, 4\}$, let $D_k \in \mathbb{R}^{n \times d}$ be defined as Definition 48.*
- *For $k_3 \in \{6, 7, 2, 4\}$, let $z_{k_3}(X) \in \mathbb{R}^{n \times n}$ be defined as Lemma 43, 44, 46, and 47, respectively.*
- *Let $K \in \mathbb{R}^n$ be defined as Definition 49.*
- *We define $z_6(X) \in \mathbb{R}^{n \times n}$, which satisfies*

$$\underbrace{z_6(X)}_{n \times n} = \underbrace{f(X)}_{n \times n} \underbrace{\mathrm{diag}(K)}_{n \times n}.$$

- *We define $z_7(X) \in \mathbb{R}^{n \times n}$, which satisfies*

$$\underbrace{z_7(X)}_{n \times n} = \underbrace{f(X)}_{n \times n} \odot (\underbrace{h(X)}_{n \times d} \underbrace{G_i^\top}_{d \times n})$$

- *We define $z_2(X) \in \mathbb{R}^{n \times n}$, which satisfies*

$$\underbrace{z_2(X)}_{n \times n} = \underbrace{\mathrm{diag}(K)}_{n \times n} \underbrace{f(X)}_{n \times n}$$

- *We define $z_4(X) \in \mathbb{R}^{n \times n}$, which satisfies*

$$\underbrace{z_4(X)}_{n \times n} = \underbrace{f(X)}_{n \times n} \odot (\underbrace{G_i}_{n \times d} \underbrace{h(X)^\top}_{d \times n})$$

*Then, we can show that the close forms of $D_k$ can be written as follows:*

- $D_6 = -\underbrace{z_6(X)}_{n \times n} \underbrace{X}_{n \times d} \underbrace{W^\top}_{d \times d}.$
- $D_7 = \underbrace{z_7(X)}_{n \times n} \underbrace{X}_{n \times d} \underbrace{W^\top}_{d \times d}.$

- $D_8 = \underbrace{f(X)}_{n \times n} \underbrace{G_i}_{n \times d} \underbrace{W_V^\top}_{d \times d}$.
- $D_2 = -\underbrace{z_2(X)}_{n \times n} \underbrace{X}_{n \times d} \underbrace{W}_{d \times d}$.
- $D_4 = \underbrace{z_4(X)}_{n \times n} \underbrace{X}_{n \times d} \underbrace{W}_{d \times d}$.

**Proof** We finish the proof by parts.

- By Lemma 43, we have the close form of $D_6$.
- By Lemma 44, we have the close form of $D_7$.
- By Lemma 45, we have the close form of $D_8$.
- By Lemma 46, we have the close form of $D_2$.
- By Lemma 47, we have the close form of $D_4$.

■

## Appendix H. Fast Computation for Gradient on $T(X)$

In this section, we give an almost linear time $n^{1+o(1)}$ algorithm for each $B_i(X)$ term. Namely, we consider $B_6(X), B_7(X), B_8(X), B_2(X), B_4(X)$ in Section H.1, H.2, H.3, H.4, and H.5, respectively.

### H.1. Fast computation for $B_6(X)$ term

Before we introduce the almost linear time algorithm for $B_6(X)$ term, we need to introduce the accelerated algorithm for the key component term, $z_6(X)$, in Lemma 52.

We first compute $K$, which is defined in Definition 49

**Lemma 51 (Computation time for $K$)** *If we have the below conditions,*

- *Let $K \in \mathbb{R}^n$ be defined as Definition 49.*

*Then, we can show that $K$ can be computed in $O(n \cdot d)$ time.*

**Proof** Since for each $i_0 \in [n]$, we have

$$\underbrace{K_{i_0}}_{1 \times 1} = \underbrace{G_i(i_0, *)^\top}_{1 \times d} \underbrace{s(X)_{i_0,*}}_{d \times 1}$$

Then, we have that it takes $O(d)$ time for calculating each entry.

Since there are total $n$ entries in $K$, the overall computation time for $K$ is $O(n \cdot d)$. ■

We now compute $z_6(X)$.

**Lemma 52 (Fast computation for $z_6(X)$)** *If we have the below conditions,*

- *Let $X \in \mathbb{R}^{n \times d}, W, W_V \in \mathbb{R}^{d \times d}$ be defined in Definition 2.*
- *Let $G_i \in \mathbb{R}^{n \times d}$ denote the gradient matrix resulting from the application of the chain rule up to the function $g_i$, i.e., $G_i = \frac{\mathrm{d}L(X)}{\mathrm{d}\mathsf{Attn}_i(T_{i-1}(X))}$.*
- *Assuming each entry of $X, W, W_V, G_i$ can be re represented using $O(\log(n))$ bits.*
- *Let $z_6(X) \in \mathbb{R}^{n \times n}$ be defined in Lemma 43.*

*Then, for some $k_6 = n^{o(1)}$, there are matrices $U_6, V_6 \in \mathbb{R}^{n \times k_6}$ such that $\|U_6 V_6^\top - z_6(X)\|_\infty \leq \epsilon / \operatorname{poly}(n)$. The matrices $U_6, V_6$ can be constructed in $n^{1+o(1)}$ time.*

**Proof** Recall in Lemma 43, we have define $z_6(X)$ satisfying the following equation

$$\underbrace{z_6(X)_{*,i_0}}_{n \times 1} = (\underbrace{G_i(i_0,*)^\top}_{1 \times d} \underbrace{s(X)_{i_0,*}}_{d \times 1}) \underbrace{f(X)_{*,i_0}}_{n \times 1} \tag{3}$$

Recall that $K \in \mathbb{R}^n$ has been defined in Definition 49. By Lemma 51, we have $K$ can be computed in $O(n \cdot d)$ time.

We also have

$$\underbrace{z_6(X)}_{n \times n} = \underbrace{f(X)}_{n \times n} \underbrace{\operatorname{diag}(K)}_{n \times n}$$

By Lemma 27, we have $U_1, V_1 \in \mathbb{R}^{n \times k_1}$ such that

$$\|U_1 V_1^\top - f(X)\|_\infty \leq \epsilon / \operatorname{poly}(n)$$

Let $U_6 = U_1$, $V_6 = \operatorname{diag}(K) V_1$.

We have $V_6 = \underbrace{\operatorname{diag}(K)}_{n \times n} \underbrace{V_1}_{n \times k_1}$ can be computed in $n k_1$ time.

The overall running time for constructing $U_6$ and $V_6$ is $n^{1+o(1)}$.

Then, we consider the error bound.

We have

$$\begin{aligned}
\|U_6 V_6^\top - z_6(X)\|_\infty &= \|U_1 V_1^\top \operatorname{diag}(K) - f(X) \operatorname{diag}(K)\|_\infty \\
&\leq n \|U_1 V_1^\top - f(X)\|_\infty \|\operatorname{diag}(K)\|_\infty \\
&\leq n(\epsilon / \operatorname{poly}(n)) \|\operatorname{diag}(K)\|_\infty \\
&\leq \epsilon / \operatorname{poly}(n)
\end{aligned}$$

where the 1st step is from the choice of $U_6$, $V_6$, the 2nd step comes from basic linear algebra, the 3rd step is because of Lemma 27, the 4th step is due to $\|\operatorname{diag}(K)\|_\infty \leq \operatorname{poly}(n)$. ■

Then, we are ready to introduce the almost linear time algorithm for $B_6(X)$ term.

**Lemma 53 (Fast computation for $B_6(X)$ term)** *If we have the below conditions,*

- *Let $X \in \mathbb{R}^{n \times d}, W, W_V \in \mathbb{R}^{d \times d}$ be defined in Definition 2.*
- *Assuming each entry of $X, W, W_V, G_i$ can be re represented using $O(\log(n))$ bits.*
- *Let $B_6(X) \in \mathbb{R}^{n \times n}$ be defined in Lemma 36.*
- *We define $D_6 \in \mathbb{R}^{n \times d}$, where $D_6 := \sum_{i_0=1}^{n} \sum_{j_0=1}^{d} G_i(i_0, j_0) B_6(X)$.*
- *Let $G_i \in \mathbb{R}^{n \times d}$ denote the gradient matrix resulting from the application of the chain rule up to the function $g_i$, i.e., $G_i = \frac{\mathrm{d}L(X)}{\mathrm{d}\mathsf{Attn}_i(T_{i-1}(X))}$.*
- *For $i_2 \in [n], j_2 \in [d]$, let $G_i(i_2, j_2)$ denote the $(i_2, j_2)$-th entry of $G_i$.*

*Then, we can show that, there is an algorithm to approximate $D_6$ in $n^{1+o(1)}$ time, and it can achieve $\epsilon/\operatorname{poly}(n)$ accuracy.*

*Namely, the algorithm output $\widetilde{D}_6$ satisfying*

$$\|D_6 - \widetilde{D}_6\|_\infty \leq \epsilon/\operatorname{poly}(n)$$

**Proof**

Recall that in Lemma 43, we have defined $z_6(X) \in \mathbb{R}^{n \times n}$, which satisfies

$$\underbrace{z_6(X)_{*,i_0}}_{n \times 1} = (\underbrace{G_i(i_0, *)^\top}_{1 \times d} \underbrace{s(X)_{i_0,*}}_{d \times 1}) \underbrace{f(X)_{*,i_0}}_{n \times 1}$$

And, in that Lemma, we also have

$$\sum_{i_0=1}^{n} \sum_{j_0=1}^{d} \underbrace{G_i(i_0, j_0)}_{1 \times 1} \underbrace{B_6(X)}_{n \times d} = - \underbrace{z_6(X)}_{n \times n} \underbrace{X}_{n \times d} \underbrace{W^\top}_{d \times d}$$

Let $U_6, V_6 \in \mathbb{R}^{n \times k_6}$ be defined as Lemma 52.

Let $\widetilde{z}_6(X) = U_6 V_6^\top$.

By Lemma 52, we have

$$\|\widetilde{z}_6(X) - z_6(X)\|_\infty \leq \epsilon/\operatorname{poly}(n) \tag{4}$$

**Proof of running time.**

We compute in the following way:

- Compute $\underbrace{V_6^\top}_{k_6 \times n} \underbrace{X}_{n \times d}$, which takes $n^{1+o(1)}$ time.
- Compute $\underbrace{V_6^\top X}_{k_6 \times d} \underbrace{W^\top}_{d \times d}$, which takes $n^{1+o(1)}$ time.
- Compute $\underbrace{U_6}_{n \times k_6} \underbrace{V_6^\top X W^\top}_{k_6 \times d}$, which takes $n^{1+o(1)}$ time.

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

**Proof of error bound.**

We have

$$\begin{aligned}
& \|\widetilde{z}_6(X)XW^\top - z_6(X)XW^\top\|_\infty \\
\leq & d \cdot n \|\widetilde{z}_6(X) - z_6(X)\|_\infty \|X\|_\infty \|W\|_\infty \\
\leq & d \cdot n (\epsilon/\operatorname{poly}(n)) \|X\|_\infty \|W\|_\infty \\
\leq & \epsilon/\operatorname{poly}(n)
\end{aligned}$$

where the 1st step is from basic linear algebra, the 2nd step comes from Eq.(4), the 3rd step is because of $\|W\|_\infty \leq \operatorname{poly}(n)$ and $\|X\|_\infty \leq \operatorname{poly}(n)$.

■

## H.2. Fast computation for $B_7(X)$ term

Similar to the analysis process of $B_6(X)$ term, we first provide the almost linear time algorithm for $z_7(X)$, then provide that algorithm for $B_7(X)$.

**Lemma 54 (Fast computation for $z_7(X)$)** *If we have the below conditions,*

- *Let $z_7(X) \in \mathbb{R}^{n \times n}$ be defined in Lemma 44.*
- *By Lemma 27, let $U_1, V_1$ be the low rank approximation of $f(X)$, such that $\|U_1 V_1^\top - f(X)\|_\infty \leq \epsilon/\operatorname{poly}(n)$.*
- *Let $X \in \mathbb{R}^{n \times d}, W, W_V \in \mathbb{R}^{d \times d}$ be defined in Definition 2.*
- *Assuming each entry of $X, W, W_V, G_i$ can be re represented using $O(\log(n))$ bits.*
- *Let $G_i \in \mathbb{R}^{n \times d}$ denote the gradient matrix resulting from the application of the chain rule up to the function $g_i$, i.e., $G_i = \frac{\mathrm{d}L(X)}{\mathrm{dAttn}_i(T_{i-1}(X))}$.*
- *For $i_2 \in [n], j_2 \in [d]$, let $G_i(i_2, j_2)$ denote the $(i_2, j_2)$-th entry of $G_i$.*

*Then, for some $k_7 = n^{o(1)}$, there are matrices $U_7, V_7 \in \mathbb{R}^{n \times k_7}$ such that $\|U_7 V_7^\top - z_7(X)\|_\infty \leq \epsilon/\operatorname{poly}(n)$. The matrices $U_7, V_7$ can be constructed in $n^{1+o(1)}$ time.*

**Proof** Recall that in Lemma 44, we have defined $z_7(X) \in \mathbb{R}^{n \times n}$, where the $i_0$-th column of $z_7(X)$ satisfies

$$\underbrace{z_7(X)_{*,i_0}}_{n \times 1} = \underbrace{f(X)_{*,i_0}}_{n \times 1} \odot (\underbrace{h(X)}_{n \times d} \underbrace{G_i(i_0, *)}_{d \times 1})$$

which is equivalent to

$$\underbrace{z_7(X)}_{n \times n} = \underbrace{f(X)}_{n \times n} \odot (\underbrace{h(X)}_{n \times d} \underbrace{G_i^\top}_{d \times n})$$

By Lemma 27, we know $\widetilde{f}(X) := U_1 V_1^\top$ is a good approximation for $f(X)$.

We choose $U_7 = U_1 \oslash h(X)$ and $V_7 = V_1 \oslash G_i$, where $U_7, V_7 \in \mathbb{R}^{n \times k_1 d}$.

**Proof of running time.**

For $U_7 = U_1 \oslash h(X)$, since $U_1 \in \mathbb{R}^{n \times k_1}, h(X) \in \mathbb{R}^{n \times d}$, constructing $U_7$ takes $O(ndk_1) = O(n^{1+o(1)})$ time.

Similarly, constructing $V_7$ takes $O(n^{1+o(1)})$ time.

**Proof of error bound.**

Using Fact 17, we have

$$\begin{aligned} \|U_7 V_7^\top - z_7(X)\|_\infty &= \|U_7 V_7^\top - f(X) \odot (h(X) G_i^\top)\|_\infty \\ &= \|(U_1 \oslash h(X))(V_1 \oslash G_i)^\top - f(X) \odot (h(X) G_i^\top)\|_\infty \\ &= \|(U_1 V_1^\top) \odot (h(X) G_i^\top) - f(X) \odot (h(X) G_i^\top)\|_\infty \\ &= \|\widetilde{f}(X) \odot (h(X) G_i^\top) - f(X) \odot (h(X) G_i^\top)\|_\infty \\ &\leq d \|h(X)\|_\infty \|G_i\|_\infty \cdot \epsilon / \operatorname{poly}(n) \\ &\leq \epsilon / \operatorname{poly}(n) \end{aligned} \tag{5}$$

where the 1st step is from the definition of $z_7(X)$, the 2nd step comes from the choice of $U_7$ and $V_7$, the 3rd step is because of Fact 17, the 4th step is due to the definition of $\widetilde{f}(X)$, the 5th step follows from $\|\widetilde{f}(X) - f(X)\|_\infty \leq \epsilon / \operatorname{poly}(n)$, the sixth step follows from Lemma 32 and $\|G_i\|_\infty \leq \operatorname{poly}(n)$.

■

Then, we can do similarly fast computation for $B_7$ term.

**Lemma 55 (Fast computation for $B_7(X)$ term)** *If we have the below conditions,*

- *Let $B_7(X) \in \mathbb{R}^{n \times d}$ be defined in Lemma 37.*
- *We define $D_7 \in \mathbb{R}^{n \times d}$, where $D_7 := \sum_{i_0=1}^n \sum_{j_0=1}^d G_i(i_0, j_0) B_7(X)$.*
- *Let $X \in \mathbb{R}^{n \times d}, W, W_V \in \mathbb{R}^{d \times d}, B \in \mathbb{R}^{n \times d}$ be defined in Definition 2.*
- *Assuming each entry of $X, W, W_V, G_i$ can be re represented using $O(\log(n))$ bits.*
- *Let $G_i \in \mathbb{R}^{n \times d}$ denote the gradient matrix resulting from the application of the chain rule up to the function $g_i$, i.e., $G_i = \frac{\mathrm{d}L(X)}{\mathrm{d}\mathsf{Attn}_i(T_{i-1}(X))}$.*
- *For $i_2 \in [n], j_2 \in [d]$, let $G_i(i_2, j_2)$ denote the $(i_2, j_2)$-th entry of $G_i$.*

*Then, we can show that, there is an algorithm to approximate $D_7$ in $n^{1+o(1)}$ time, and it can achieve $\epsilon / \operatorname{poly}(n)$ accuracy.*

*Namely, the algorithm output $\widetilde{D}_7$ satisfies*

$$\|D_7 - \widetilde{D}_7\|_\infty \leq \epsilon / \operatorname{poly}(n)$$

**Proof** In Lemma 44, we have

$$\sum_{i_0=1}^{n}\sum_{j_0=1}^{d} \underbrace{G_i(i_0,j_0)}_{1\times 1} \underbrace{B_7(X)}_{n\times d} = \underbrace{z_7(X)}_{n\times n} \underbrace{X}_{n\times d} \underbrace{W^\top}_{d\times d}$$

Let $U_7, V_7 \in \mathbb{R}^{n\times k_7}$ be defined in Lemma 54.
Let $\widetilde{z}_7(X) := U_7 V_7^\top$.
By Lemma 54, we have

$$\|\widetilde{z}_7(X) - z_7(X)\|_\infty \le \epsilon/\operatorname{poly}(n) \tag{6}$$

**Proof of running time.**
We compute in the following way:

- Compute $\underbrace{V_7^\top}_{k_7\times n} \underbrace{X}_{n\times d}$, which takes $n^{1+o(1)}$ time.
- Compute $\underbrace{V_7^\top X}_{k_7\times d} \underbrace{W^\top}_{d\times d}$, which takes $n^{1+o(1)}$ time.
- Compute $\underbrace{U_7}_{n\times k_7} \underbrace{V_7^\top X W^\top}_{k_7\times d}$, which takes $n^{1+o(1)}$ time.

Therefore, the overall running time is $n^{1+o(1)}$.
**Proof of error bound.**
We have

$$\begin{aligned}
& \|\widetilde{z}_7(X) X W^\top - z_7(X) X W^\top\|_\infty \\
\le & \ d \cdot n \|\widetilde{z}_7(X) - z_7(X)\|_\infty \|X\|_\infty \|W\|_\infty \\
\le & \ d \cdot n (\epsilon/\operatorname{poly}(n)) \|X\|_\infty \|W\|_\infty \\
\le & \ \epsilon/\operatorname{poly}(n)
\end{aligned}$$

where the 1st step is from basic linear algebra, the 2nd step comes from Eq. (6), the 3rd step is because of $\|W\|_\infty \le \operatorname{poly}(n)$ and $\|X\|_\infty \le \operatorname{poly}(n)$.

■

### H.3. Fast computation for $B_8(X)$ term

Then, we can do fast computations on $B_8(X)$ term.

**Lemma 56 (Fast computation for $B_8(X)$ term)** *If we have the below conditions,*

- *Let $B_8(X) \in \mathbb{R}^{n\times d}$ be defined in Lemma 38.*
- *We define $D_8 \in \mathbb{R}^{n\times d}$, where $D_8 := \sum_{i_0=1}^{n}\sum_{j_0=1}^{d} G_i(i_0,j_0) B_8(X)$.*
- *Let $X \in \mathbb{R}^{n\times d}, W, W_V \in \mathbb{R}^{d\times d}$ be defined in Definition 2.*

- *Assuming each entry of $X, W, W_V, G_i$ can be re represented using $O(\log(n))$ bits.*
- *Let $G_i \in \mathbb{R}^{n \times d}$ denote the gradient matrix resulting from the application of the chain rule up to the function $g_i$, i.e., $G_i = \frac{\mathrm{d}L(X)}{\mathrm{d}\mathsf{Attn}_i(T_{i-1}(X))}$.*
- *For $i_2 \in [n], j_2 \in [d]$, let $G_i(i_2, j_2)$ denote the $(i_2, j_2)$-th entry of $G_i$.*

*Then, we can show that, there is an algorithm to approximate $D_8$ in $n^{1+o(1)}$ time, and it can achieve $\epsilon/\operatorname{poly}(n)$ accuracy.*

*Namely, the algorithm output $\widetilde{D}_8$ satisfies*

$$\|D_8 - \widetilde{D}_8\|_\infty \leq \epsilon/\operatorname{poly}(n)$$

**Proof** Recall that in Lemma 45, we have

$$\sum_{i_0=1}^{n} \sum_{j_0=1}^{d} \underbrace{G_i(i_0, j_0)}_{1 \times 1} \underbrace{B_8(X)}_{n \times d} = \underbrace{f(X)}_{n \times n} \underbrace{G_i}_{n \times d} \underbrace{W_V^\top}_{d \times d}$$

Let $\widetilde{f}(X) := U_1 V_1^\top$ denote the approximation of $f(X)$.

By Lemma 27, we have

$$\|f(X) - \widetilde{f}(X)\|_\infty \leq \epsilon/\operatorname{poly}(n) \tag{7}$$

**Proof of running time.**

We compute in the following way:

- Compute $\underbrace{V_1^\top}_{k_1 \times n} \underbrace{G_i}_{n \times d}$, which takes $n^{1+o(1)}$ time.
- Compute $\underbrace{V_1^\top G_i}_{k_1 \times d} \underbrace{W_V^\top}_{d \times d}$, which takes $n^{1+o(1)}$ time.
- Compute $\underbrace{U_1}_{n \times k_1} \underbrace{V_1^\top G_i W_V^\top}_{k_1 \times d}$, which takes $n^{1+o(1)}$ time.

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

**Proof of error bound.**

We have

$$\begin{aligned}
&\|\widetilde{f}(X) G_i W_V^\top - f(X) G_i W_V^\top\|_\infty \\
&\leq d \cdot n \|\widetilde{f}(X) - f(X)\|_\infty \|G_i\|_\infty \|W_V\|_\infty \\
&\leq d \cdot n (\epsilon/\operatorname{poly}(n)) \|G_i\|_\infty \|W_V\|_\infty \\
&\leq \epsilon/\operatorname{poly}(n)
\end{aligned}$$

where the 1st step is from basic linear algebra, the 2nd step comes from Eq.(7), the 3rd step is because of $\|G_i\|_\infty \leq \operatorname{poly}(n)$ and $\|W_V\|_\infty \leq \operatorname{poly}(n)$.

■

### H.4. Fast computation for $B_2(X)$ term

Then, we provide the proof of how to do fast computation on $B_2(X)$.

**Lemma 57 (Fast computation for $z_2(X)$)** *If we have the below conditions,*

- *Let $z_2(X) \in \mathbb{R}^{n \times n}$ be defined as in Lemma 46.*
- *Let $X \in \mathbb{R}^{n \times d}, W, W_V \in \mathbb{R}^{d \times d}$ be defined in Definition 2.*
- *Assuming each entry of $X, W, W_V, G_i$ can be re represented using $O(\log(n))$ bits.*
- *Let $G_i \in \mathbb{R}^{n \times d}$ denote the gradient matrix resulting from the application of the chain rule up to the function $g_i$, i.e., $G_i = \frac{\mathrm{d}L(X)}{\mathrm{d}\mathsf{Attn}_i(T_{i-1}(X))}$.*
- *For $i_2 \in [n], j_2 \in [d]$, let $G_i(i_2, j_2)$ denote the $(i_2, j_2)$-th entry of $G_i$.*

*Then, for some $k_9 = n^{o(1)}$, there are matrices $U_9, V_9 \in \mathbb{R}^{n \times k_9}$ such that $\|U_9 V_9^\top - z_2(X)\|_\infty \leq \epsilon / \operatorname{poly}(n)$. The matrices $U_9, V_9$ can be constructed in $n^{1+o(1)}$ time.*

**Proof**

Recall that in Lemma 46, we have defined $z_2(X) \in \mathbb{R}^{n \times n}$, where the $i_0$-th row of $z_2(X)$ satisfies

$$\underbrace{z_2(X)_{i_0,*}}_{n \times 1} = (\underbrace{G_i(i_0,*)^\top}_{1 \times d} \underbrace{s(X)_{i_0,*}}_{d \times 1}) \underbrace{f(X)_{i_0,*}}_{n \times 1}$$

Recall that $K \in \mathbb{R}^n$ has been defined in Definition 49.

By Lemma 51, we have $K$ can be computed in $O(n \cdot d)$ time.

We also have

$$\underbrace{z_2(X)}_{n \times n} = \underbrace{\operatorname{diag}(K)}_{n \times n} \underbrace{f(X)}_{n \times n}$$

By Lemma 27, let $U_1, V_1$ be the low rank approximation of $f(X)$, such that $\|U_1 V_1^\top - f(X)\|_\infty \leq \epsilon / \operatorname{poly}(n)$.

Let $U_9 = \operatorname{diag}(K) U_1$, $V_6 = V_1$.

We have $U_9 = \underbrace{\operatorname{diag}(K)}_{n \times n} \underbrace{U_1}_{n \times k_1}$ can be computed in $nk_1$ time.

The overall running time for constructing $U_9$ and $V_9$ is $n^{1+o(1)}$.

Then, we consider the error bound.

We have

$$\begin{aligned}
\|U_9 V_9^\top - z_2(X)\|_\infty &= \|\operatorname{diag}(K) U_1 V_1^\top - \operatorname{diag}(K) f(X)\|_\infty \\
&\leq n \|U_1 V_1^\top - f(X)\|_\infty \|\operatorname{diag}(K)\|_\infty \\
&\leq n(\epsilon / \operatorname{poly}(n)) \|\operatorname{diag}(K)\|_\infty \\
&\leq \epsilon / \operatorname{poly}(n) \qquad (8)
\end{aligned}$$

where the 1st step is from the choice of $U_6$, $V_6$, the 2nd step comes from basic linear algebra, the 3rd step is because of Lemma 27, the 4th step is due to $\|\operatorname{diag}(K)\|_\infty \leq \operatorname{poly}(n)$.

■

**Lemma 58 (Fast computation for $B_2(X)$ term)** *If we have the below conditions,*

- *Let $B_2(X) \in \mathbb{R}^{n \times d}$ be defined in Lemma 39.*
- *We define $D_2 \in \mathbb{R}^{n \times d}$, where $D_2 := \sum_{i_0=1}^{n} \sum_{j_0=1}^{d} \underbrace{G_i(i_0, j_0)}_{1 \times 1} \underbrace{e_{i_0}}_{n \times 1} \underbrace{B_2(X)^\top}_{1 \times d}$.*
- *Let $X \in \mathbb{R}^{d \times n}, W, W_V \in \mathbb{R}^{d \times d}, B \in \mathbb{R}^{n \times d}$ be defined in Definition 2.*
- *Assuming each entry of $X, W, W_V, B, G_i$ can be re represented using $O(\log(n))$ bits.*
- *Let $G_i \in \mathbb{R}^{n \times d}$ denote the gradient matrix resulting from the application of the chain rule up to the function $g_i$, i.e., $G_i = \frac{\mathrm{d}L(X)}{\mathrm{d}\mathsf{Attn}_i(T_{i-1}(X))}$.*
- *For $i_2 \in [n], j_2 \in [d]$, let $G_i(i_2, j_2)$ denote the $(i_2, j_2)$-th entry of $G_i$.*

*Then, we can show that, there is an algorithm to approximate $D_2$ in $n^{1+o(1)}$ time, and it can achieve $\epsilon/\operatorname{poly}(n)$ accuracy.*

*Namely, the algorithm output $\widetilde{D}_2$ satisfies*

$$\|D_2 - \widetilde{D}_2\|_\infty \leq \epsilon/\operatorname{poly}(n)$$

**Proof**

In Lemma 46, we have

$$\sum_{i_0=1}^{n} \sum_{j_0=1}^{d} \underbrace{G_i(i_0, j_0)}_{1 \times 1} \underbrace{e_{i_0}}_{n \times 1} \underbrace{B_2(X)^\top}_{1 \times d} = -\underbrace{z_2(X)}_{n \times n} \underbrace{X}_{n \times d} \underbrace{W}_{d \times d}$$

Let $U_9, V_9 \in \mathbb{R}^{n \times k_9}$ be defined in Lemma 57.
Let $\widetilde{z}_2(X) := U_9 V_9^\top$.
By Lemma 57, we have

$$\|\widetilde{z}_2(X) - z_2(X)\|_\infty \leq \epsilon/\operatorname{poly}(n) \tag{9}$$

**Proof of running time.**

We compute in the following way:

- Compute $\underbrace{V_9^\top}_{k_9 \times n} \underbrace{X}_{n \times d}$, which takes $n^{1+o(1)}$ time.
- Compute $\underbrace{V_9^\top X}_{k_9 \times d} \underbrace{W}_{d \times d}$, which takes $n^{1+o(1)}$ time.
- Compute $\underbrace{U_9}_{n \times k_9} \underbrace{V_9^\top X W}_{k_9 \times d}$, which takes $n^{1+o(1)}$ time.

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

**Proof of error bound.**

We have

$$\begin{aligned} & \|\widetilde{z}_2(X)XW - z_2(X)XW\|_\infty \\ \leq & d \cdot n \|\widetilde{z}_2(X) - z_2(X)\|_\infty \|X\|_\infty \|W\|_\infty \\ \leq & d \cdot n(\epsilon/\operatorname{poly}(n)) \|X\|_\infty \|W\|_\infty \\ \leq & \epsilon/\operatorname{poly}(n) \end{aligned}$$

where the 1st step is from basic linear algebra, the 2nd step comes from Eq.(9), the 3rd step is because of $\|W\|_\infty \leq \operatorname{poly}(n)$ and $\|X\|_\infty \leq \operatorname{poly}(n)$.

■

### H.5. Fast computation for $B_4(X)$ term

Finally, our analysis shows that we can do fast computations for $B_4(X)$ term. After that, we showed that all terms can be computed quickly.

**Lemma 59 (Fast computation for $z_4(X)$)** *If we have the below conditions,*

- *Let $z_4(X) \in \mathbb{R}^{n \times n}$ be defined in Lemma 47.*
- *Let $X \in \mathbb{R}^{n \times d}, W, W_V \in \mathbb{R}^{d \times d}$ be defined in Definition 2.*
- *Assuming each entry of $X, W, W_V, G_i$ can be re represented using $O(\log(n))$ bits.*
- *Let $G_i \in \mathbb{R}^{n \times d}$ denote the gradient matrix resulting from the application of the chain rule up to the function $g_i$, i.e., $G_i = \frac{\mathrm{d}L(X)}{\mathrm{d}\mathsf{Attn}_i(T_{i-1}(X))}$.*
- *For $i_2 \in [n], j_2 \in [d]$, let $G_i(i_2, j_2)$ denote the $(i_2, j_2)$-th entry of $G_i$.*

*Then, for some $k_{10} = n^{o(1)}$, there are matrices $U_{10}, V_{10} \in \mathbb{R}^{n \times k_{10}}$, let $\widetilde{z}_4(X) := U_{10} V_{10}^\top$, such that $\|\widetilde{z}_4(X) - z_4(X)\|_\infty \leq \epsilon/\operatorname{poly}(n)$. The matrices $U_{10}, V_{10}$ can be constructed in $n^{1+o(1)}$ time.*

**Proof**

In Lemma 47, we have defined $z_4(X) \in \mathbb{R}^{n \times n}$, where the $i_0$-th column of $z_4(X)$ satisfies

$$\underbrace{z_4(X)_{i_0,*}}_{n \times 1} = (\underbrace{f(X)_{i_0,*}}_{n \times 1} \odot \underbrace{(h(X) G_i(i_0, *))}_{n \times 1})$$

which is equivalent to

$$\underbrace{z_4(X)}_{n \times n} = (\underbrace{f(X)}_{n \times n} \odot \underbrace{G_i}_{n \times d} \underbrace{h(X)^\top}_{d \times n})$$

By Lemma 27, let $U_1, V_1$ be the low rank approximation of $f(X)$, such that $\|U_1 V_1^\top - f(X)\|_\infty \leq \epsilon/\operatorname{poly}(n)$.

We choose $U_{10} = U_1 \oslash G_i$ and $V_{10} = V_1 \oslash h(X)$, where $U_{10}, V_{10} \in \mathbb{R}^{n \times k_1 d}$.

**Proof of running time.**

For $U_{10} = U_1 \oslash G_i$, since $U_1 \in \mathbb{R}^{n \times k_1}, G_i \in \mathbb{R}^{n \times d}$, constructing $U_{10}$ takes $O(ndk_1) = O(n^{1+o(1)})$ time.

Similarly, constructing $V_{10}$ takes $O(n^{1+o(1)})$ time.

**Proof of error bound.**

Let $\widetilde{f}(X) := U_1 V_1^\top$.

Using Fact 17, we have

$$\begin{aligned} & \|\widetilde{z}_4(X) - z_4(X)\|_\infty \\ = & \|U_{10} V_{10}^\top - f(X) \odot (G_i \cdot h(X)^\top)\|_\infty \\ = & \|(U_1 \oslash G_i)(V_1 \oslash h(X))^\top - f(X) \odot (G_i \cdot h(X)^\top)\|_\infty \\ = & \|(U_1 V_1^\top) \odot (G_i \cdot h(X)^\top) - f(X) \odot (G_i \cdot h(X)^\top)\|_\infty \end{aligned}$$

where the 1st step is from the definition of $\widetilde{z}_4(X), z_4(X)$, the 2nd step comes from the choice of $U_{10}$ and $V_{10}$, the 3rd step is because of Fact 17.

$$\begin{aligned} & \|(U_1 V_1^\top) \odot (G_i \cdot h(X)^\top) - f(X) \odot (G_i \cdot h(X)^\top)\|_\infty \\ = & \|U_1 V_1^\top - f(X)\|_\infty \|G_i \cdot h(X)^\top\|_\infty \\ \leq & d \cdot (\epsilon / \operatorname{poly}(n)) \|h(X)\|_\infty \|G_i\|_\infty \\ \leq & \epsilon / \operatorname{poly}(n) \end{aligned}$$

where the 1st step is from basic linear algebra, the 2nd step comes from $\|U_1 V_1 - f(X)\|_\infty \leq \epsilon / \operatorname{poly}(n)$, the 3rd step is because of Lemma 32 and $\|G_i\|_\infty \leq \operatorname{poly}(n)$.

■

**Lemma 60 (Fast computation for $B_4(X)$ term)** *If we have the below conditions,*

- *Let $B_4(X) \in \mathbb{R}^{n \times d}$ be defined in Lemma 40.*
- *We define $D_4 \in \mathbb{R}^{n \times d}$, where $D_4 := \sum_{i_0=1}^n \sum_{j_0=1}^d \underbrace{G_i(i_0, j_0)}_{1 \times 1} \underbrace{e_{i_0}}_{n \times 1} \underbrace{B_4(X)^\top}_{1 \times d}$.*
- *Let $X \in \mathbb{R}^{n \times d}, W, W_V \in \mathbb{R}^{d \times d}$ be defined in Definition 2.*
- *Assuming each entry of $X, W, W_V, G_i$ can be re represented using $O(\log(n))$ bits.*
- *Let $G_i \in \mathbb{R}^{n \times d}$ denote the gradient matrix resulting from the application of the chain rule up to the function $g_i$, i.e., $G_i = \frac{\mathrm{d} L(X)}{\mathrm{d} \mathsf{Attn}_i(T_{i-1}(X))}$.*
- *For $i_2 \in [n], j_2 \in [d]$, let $G_i(i_2, j_2)$ denote the $(i_2, j_2)$-th entry of $G_i$.*

*Then, we can show that, there is an algorithm to approximate $D_4$ in $n^{1+o(1)}$ time, and it can achieve $\epsilon / \operatorname{poly}(n)$ accuracy.*

*Namely, the algorithm output $\widetilde{D}_4$ satisfies*

$$\|D_4 - \widetilde{D}_4\|_\infty \leq \epsilon / \operatorname{poly}(n)$$

**Proof** In Lemma 47, we have

$$\sum_{i_0=1}^{n}\sum_{j_0=1}^{d}\underbrace{G_i(i_0,j_0)}_{1\times 1}\underbrace{e_{i_0}}_{n\times 1}\underbrace{B_4(X)^\top}_{1\times d}=\underbrace{z_4(X)}_{n\times n}\underbrace{X}_{n\times d}\underbrace{W}_{d\times d}$$

Let $\widetilde{z}_4(X) := U_{10}V_{10}^\top$.
By Lemma 59, we have

$$\|\widetilde{z}_4(X)-z_4(X)\|_\infty \leq \epsilon/\operatorname{poly}(n) \tag{10}$$

**Proof of running time.**
We compute in the following way:

- Compute $\underbrace{V_{10}^\top}_{k_{10}\times n}\underbrace{X}_{n\times d}$, which takes $n^{1+o(1)}$ time.
- Compute $\underbrace{V_{10}^\top X}_{k_{10}\times d}\underbrace{W}_{d\times d}$, which takes $n^{1+o(1)}$ time.
- Compute $\underbrace{U_{10}}_{n\times k_{10}}\underbrace{V_{10}^\top XW}_{k_{10}\times d}$, which takes $n^{1+o(1)}$ time.

Therefore, the overall running time is $n^{1+o(1)}$.
**Proof of error bound.**
We have

$$\begin{aligned}
& \|\widetilde{z}_4(X)XW - z_4(X)XW\|_\infty \\
\leq & \, d\cdot n\|\widetilde{z}_4(X)-z_4(X)\|_\infty\|X\|_\infty\|W\|_\infty \\
\leq & \, d\cdot n(\epsilon/\operatorname{poly}(n))\|X\|_\infty\|W\|_\infty \\
\leq & \, \epsilon/\operatorname{poly}(n)
\end{aligned}$$

where the 1st step is from basic linear algebra, the 2nd step comes from Eq.(10), the 3rd step is because of $\|W\|_\infty \leq \operatorname{poly}(n)$ and $\|X\|_\infty \leq \operatorname{poly}(n)$.

■

### H.6. Putting everything together

After we have analyzed each $B_i(X)$ term in the previous section, we put them together in this section, to analyze the overall running time and error bound of the gradient of $L(X)$ on $T_i(X)$ in Lemma 61.

**Lemma 61 (Fast computation for $\frac{\mathrm{d}L(X)}{\mathrm{d}T_{i-1}(X)}$, formal version of Lemma 11)** *If we have the below conditions,*

- *Let $L(X)$ be defined as Definition 5.*

- *Let $m$ denote the number of self-attention transformer model (see Definition 3).*
- *For any $i \in [m]$, let $T_i(X)$ be defined as Definition 6.*
- *Let $X \in \mathbb{R}^{n \times d}, W, W_V \in \mathbb{R}^{d \times d}$ be defined in Definition 2.*
- *Assuming each entry of $X, W, W_V, G_i$ can be re represented using $O(\log(n))$ bits.*
- *Let $G_i \in \mathbb{R}^{n \times d}$ denote the gradient matrix resulting from the application of the chain rule up to the function $g_i$, i.e., $G_i = \frac{\mathrm{d}L(X)}{\mathrm{d}\mathsf{Attn}_i(T_{i-1}(X))}$.*
- *Assume $G_i$ can be computed in $n^{1+o(1)}$ time.*

*We can show that $\frac{\mathrm{d}L(X)}{\mathrm{d}T_{i-1}(X)}$ can be approximated in $n^{1+o(1)}$ time, with $1/\operatorname{poly}(n)$ approximation error. Namely, our algorithm can output $\widetilde{g}_t$ in $n^{1+o(1)}$ time, which satisfies*

$$\|\widetilde{g}_t - \frac{\mathrm{d}L(X)}{\mathrm{d}T_{i-1}(X)}\|_\infty \leq 1/\operatorname{poly}(n)$$

**Proof** By Lemma 42, we have

$$\begin{aligned} \frac{\mathrm{d}L(X)}{\mathrm{d}T_{i-1}(X)} &= \sum_{i_0=1}^{n} \sum_{j_0=1}^{d} \underbrace{G_i(i_0, j_0)}_{1 \times 1} \cdot (\underbrace{B_6(X) + B_7(X) + B_8(X)}_{n \times d} + \underbrace{e_{i_0}}_{n \times 1} \underbrace{(B_2(X) + B_4(X))^\top}_{1 \times d}) \\ &= \sum_{i \in \{2,4,6,7,8\}} D_i \end{aligned}$$

where the 1st step is from Lemma 42, the 2nd step comes from the definition of $D_6, D_7, D_8, D_2, D_4$.

Then, by Lemma 53, 55, 56, 58, 60, we have $D_6, D_7, D_8, D_2, D_4 \in \mathbb{R}^{n \times d}$ can be approximated in $n^{1+o(1)}$ time, with up to $\epsilon/\operatorname{poly}(n)$ error.

Namely, for $i \in \{2, 4, 6, 7, 8\}$, let $\widetilde{D}_i \in \mathbb{R}^{n \times d}$ denote the approximated version of $D$, we have

$$\|\widetilde{D}_i - D\|_\infty \leq \epsilon/\operatorname{poly}(n)$$

Let $\widetilde{g}_t = \sum_{i \in \{2,4,6,7,8\}} \widetilde{D}_i$.

**Proof of running time.**

The running time for $\widetilde{g}_t = \sum_{i \in \{2,4,6,7,8\}} \widetilde{D}_i$ is $5nd$.

Therefore, the overall running time for computing $\widetilde{g}_t$ is $n^{1+o(1)}$.

**Proof of error bound.**

We have

$$\begin{aligned} \|\widetilde{g}_t - \frac{\mathrm{d}L(X)}{\mathrm{d}T_{i-1}(X)}\|_\infty &= \| \sum_{i \in \{2,4,6,7,8\}} (\widetilde{D}_i - D_i)\|_\infty \\ &\leq \sum_{i \in \{2,4,6,7,8\}} \|(\widetilde{D}_i - D_i)\|_\infty \\ &\leq \epsilon/\operatorname{poly}(n) \end{aligned}$$

where the 1st step is from the definition of $\widetilde{g}_t$ and $\frac{\mathrm{d}L(X)}{\mathrm{d}T_{i-1}(X)}$, the 2nd step comes from basic algebra, the 3rd step is because of $\|\widetilde{D}_i - D\|_\infty \leq \epsilon / \mathrm{poly}(n)$.

Then, choose $\epsilon = 1/\mathrm{poly}(n)$, we have

$$\|\widetilde{g}_t - \frac{\mathrm{d}L(X)}{\mathrm{d}T_{i-1}(X)}\|_\infty \leq 1/\mathrm{poly}(n)$$

■

## Appendix I. Fast Computation for Gradient on $W$

In Section I.1, we introduce some essential notations used in this section. In Section I.2, we offer the gradient of $s(X)$ on $W$, which is equivalent to the gradient of the output of the attention mechanism on $W$. In Section I.3, we illustrate the gradient of $L(X)$ on $W$. In Section I.4, we introduce the almost linear time algorithm for calculating the gradient of $L(X)$ on $W$, along with the error bound analysis.

### I.1. Key concepts

**Definition 62 (Definition of $\mathsf{A}$, [4])** *Let $A_1, A_2 \in \mathbb{R}^{n \times d}$ be two matrices. Suppose that $\mathsf{A} = A_1 \otimes A_2 \in \mathbb{R}^{n^2 \times d^2}$. We define $\mathsf{A}_{j_0} \in \mathbb{R}^{n \times d^2}$ be a $n \times d^2$ size sub-block from $\mathsf{A}$. Note that there are $n$ such sub-blocks.*

**Remark 63** *Note that the $A_1, A_2$ matrices in Definition 62 is $X$ in our setting. Since in Alman and Song [4], they consider a more general setting, where $A_1, A_2$ can be difference matrices, while in our problem, we consider self-attention. Therefore, in our paper, we have $A_1 = A_2 = X$.*

### I.2. Gradient of $s(X)$ on $W$

We begin with introducing the close form of the gradient of $s(X)$.

Alman and Song [4] proved the close form of the gradient of $c(X) = s(X) - B$ with respect to $W$ for a constant matrix $B$. By chain rule, this is equivalent to the gradient of $s(X)$ with respect to $W$.

**Lemma 64 (Gradient of $s(X)$ on $W$, Lemma B.1 in Alman and Song [4])** *If we have the below conditions,*

- *Let $\mathsf{A}$ be defined as Definition 62. For every $i \in [d^2]$, define $\mathsf{A}_{j_0,i} \in \mathbb{R}^n$ to be the $i$-th column for $\mathsf{A}_{j_0} \in \mathbb{R}^{n \times d^2}$.*
- *Let $f(X), h(X), s(X)$ be defined as Definition 22, 23, 24.*
- *Let $W \in \mathbb{R}^{d \times d}$ be defined as Definition 2. Let $w \in \mathbb{R}^{d^2}$ denote the vector representation of $W$.*

*Then, for each $i \in [d^2]$, we have For each $j_0 \in [n]$, for every $i_0 \in [d]$*

$$\frac{\mathrm{d}s(X)_{j_0,i_0}}{\mathrm{d}w_i} = \langle \mathsf{A}_{j_0,i} \odot f(X)_{j_0}, h(X)_{i_0} \rangle - \langle f(X)_{j_0}, h(X)_{i_0} \rangle \cdot \langle \mathsf{A}_{j_0,i}, f(X)_{j_0} \rangle$$

### I.3. Gradient of $L(X)$ on $W$

Differing from the $\ell_2$ loss function used in Alman and Song [4], our framework supports arbitrary loss functions. Therefore, we use Lemma 65 to illustrate the gradient of $L(X)$ on $W$.

**Lemma 65 (Gradient of $L(X)$ on $W$)** *If we have the below conditions,*

- *Let $L(X)$ be defined as Definition 5.*
- *Let $W \in \mathbb{R}^{d \times d}, X \in \mathbb{R}^{n \times d}$ be Defined as Definition 2.*
- *Let $p(X)$ be defined as Definition 26.*

*Then, we can show that*

$$\frac{\mathrm{d}L(X)}{\mathrm{d}W_i} = X^\top \cdot p(X) \cdot X$$

**Proof** By Lemma 64, we have, for each $i \in [d^2]$, we have For each $j_0 \in [n]$, for every $i_0 \in [d]$

$$\frac{\mathrm{d}s(X)_{j_0,i_0}}{\mathrm{d}w_i} = \langle \underbrace{\mathsf{A}_{j_0,i}}_{n\times 1} \odot \underbrace{f(X)_{j_0}}_{n\times 1}, \underbrace{h(X)_{i_0}}_{n\times 1} \rangle - \langle \underbrace{f(X)_{j_0}}_{n\times 1}, \underbrace{h(X)_{i_0}}_{n\times 1} \rangle \cdot \langle \underbrace{\mathsf{A}_{j_0,i}}_{n\times 1}, \underbrace{f(X)_{j_0}}_{n\times 1} \rangle \tag{11}$$

By Fact 16, we have

$$\langle \mathsf{A}_{j_0,i} \odot f(X)_{j_0}, h(X)_{i_0} \rangle = \mathsf{A}_{j_0,i}^\top \operatorname{diag}(f(X)_{j_0}) h(X)_{i_0}$$

and

$$\langle f(X)_{j_0}, h(X)_{i_0} \rangle \cdot \langle f(X)_{j_0}, \mathsf{A}_{j_0,i} \rangle = \mathsf{A}_{j_0,i}^\top f(X)_{j_0} f(X)_{j_0}^\top h(X)_{i_0}$$

By Eq. (11), for each $i \in [d^2]$, we have For each $j_0 \in [n]$, for every $i_0 \in [d]$, we have

$$\frac{\mathrm{d}s(X)_{j_0,i_0}}{\mathrm{d}w_i} = \mathsf{A}_{j_0,i}^\top (\operatorname{diag}(f(X)_{j_0}) - f(X)_{j_0} f(X)_{j_0}^\top) h(X)_{i_0}$$

which implies,

$$\frac{\mathrm{d}s(X)_{j_0,i_0}}{\mathrm{d}W} = \underbrace{\mathsf{A}_{j_0}^\top}_{d^2\times n} \underbrace{(\operatorname{diag}(f(X)_{j_0}) - f(X)_{j_0} f(X)_{j_0}^\top)}_{n\times n} \underbrace{h(X)_{i_0}}_{n\times 1} \tag{12}$$

By Lemma 18, for $i \in [m]$, we have

$$\frac{\mathrm{d}L(X)}{\mathrm{d}W_i} = \sum_{i_2=1}^{n} \sum_{j_2=1}^{d} G_i(i_2, j_2) \cdot \frac{\mathrm{d}\mathsf{Attn}_i(T_{i-1}(X))_{i_2,j_2}}{\mathrm{d}W_i}. \tag{13}$$

By the definition of $s(X)$ (Definition 24), we have

$$s(X) = \mathsf{Attn}_i(T_{i-1}(X))$$

Combining Eq. (12) and Eq. (13), for each $i \in [m]$, we have

$$\frac{\mathrm{d}L(X)}{\mathrm{d}W_i} = \sum_{j_0=1}^{n} \sum_{i_0=1}^{d} \underbrace{G_i(j_0, i_0)}_{1\times 1} \cdot \underbrace{\mathsf{A}_{j_0}^\top}_{d^2\times n} \underbrace{(\mathrm{diag}(f(X)_{j_0}) - f(X)_{j_0} f(X)_{j_0}^\top)}_{n\times n} \underbrace{h(X)_{i_0}}_{n\times 1} \tag{14}$$

Recall that we have defined $q(X)$ in Definition 25,

$$q(X)_{j_0} := \sum_{i_0=1}^{d} G_i(j_0, i_0) \cdot h(X)_{i_0} \tag{15}$$

Recall that $p(x)_{j_0} \in \mathbb{R}^n$ is define as Definition 26,

$$p(x)_{j_0} := (\mathrm{diag}(f(x)_{j_0}) - f(x)_{j_0} f(x)_{j_0}^\top) q(x)_{j_0}. \tag{16}$$

Then, we have

$$\begin{aligned}
& \frac{\mathrm{d}L(X)}{\mathrm{d}W_i} \\
= & \sum_{j_0=1}^{n} \sum_{i_0=1}^{d} \underbrace{G_i(j_0, i_0)}_{1\times 1} \cdot \underbrace{\mathsf{A}_{j_0}^\top}_{d^2\times n} \underbrace{(\mathrm{diag}(f(X)_{j_0}) - f(X)_{j_0} f(X)_{j_0}^\top)}_{n\times n} \underbrace{h(X)_{i_0}}_{n\times 1} \\
= & \sum_{j_0=1}^{n} \underbrace{\mathsf{A}_{j_0}^\top}_{d^2\times n} \underbrace{(\mathrm{diag}(f(X)_{j_0}) - f(X)_{j_0} f(X)_{j_0}^\top)}_{n\times n} \underbrace{q(X)_{j_0}}_{n\times 1} \\
= & \sum_{j_0=1}^{n} \mathsf{A}_{j_0}^\top p_{j_0}(X) \\
= & \underbrace{X^\top}_{d\times n} \underbrace{p(X)}_{n\times n} \underbrace{X}_{n\times d}
\end{aligned}$$

where the 1st step is from Eq. (14), the 2nd step comes from Eq. (15), the 3rd step is because of Eq. (16), the 4th step is due to the tensor tricks.

■

### I.4. Fast computation

Finally, we introduce the almost linear time algorithm and its error analysis of the gradient of $L(X)$ on $W$ in Lemma 66.

**Lemma 66 (Fast computation for $\frac{\mathrm{d}L(X)}{\mathrm{d}W_i}$)** *If we have the below conditions,*

- *Let $L(X)$ be defined as Definition 5.*
- *Let $m$ denote the number of self-attention transformer layers (see Definition 3).*
- *For any $i \in [m]$, let $W_i = W_{Q_i} W_{K_i}^\top$ denote the attention weight in the $i$-th transformer layer.*

*We can show that $\frac{\mathrm{d}L(X)}{\mathrm{d}W_i}$ can be approximated in $n^{1+o(1)}$ time, with $1/\operatorname{poly}(n)$ approximation error. Namely, our algorithm can output $\widetilde{g}_w$ in $n^{1+o(1)}$ time, which satisfies*

$$\|\widetilde{g}_w - \frac{\mathrm{d}L(X)}{\mathrm{d}W_i}\|_\infty \leq 1/\operatorname{poly}(n)$$

**Proof** Recall by Lemma 29, 30, we have defined $p_1(X), p_2(X) \in \mathbb{R}^{n \times n}$.

In those Lemmas, we have $p_1(X), p_2(X)$ have low rank approximation $U_3 V_3^\top$ and $U_4 V_4^\top$, respectively.

By the definition of $p(X)$ (Definition 26), we have

$$p(X) = p_1(X) - p_2(X) \tag{17}$$

Then, by Lemma 65, we have

$$\begin{aligned} & \frac{\mathrm{d}L(X)}{\mathrm{d}W_i} \\ = & X^\top p(X) X \\ = & X^\top (p_1(X) - p_2(X)) X \end{aligned}$$

where the 1st step is from Lemma 65, the 2nd step comes from Eq. (17).

Let $\widetilde{p}_1(X), \widetilde{p}_2(X)$ denote the low rank approximations for $p_1(X), p_2(X)$, respectively.

**Proof of running time.** We first compute $X^\top \widetilde{p}_1(X) X$ in following order

- Compute $\underbrace{X^\top}_{d \times n} \underbrace{U_3}_{n \times k_3}$, which takes $n^{1+o(1)}$ time.
- Compute $\underbrace{X^\top U_3}_{d \times k_3} \underbrace{V_3^\top}_{k_3 \times n}$, which takes $n^{1+o(1)}$ time.
- Compute $\underbrace{X^\top U_3 V_3^\top}_{d \times n} \underbrace{X}_{n \times d}$, which takes $n^{1+o(1)}$ time.

The overall running time for $X^\top \widetilde{p}_1(X) X$ is $n^{1+o(1)}$.

Similarly, the overall running time for $X^\top \widetilde{p}_2(X) X$ is $n^{1+o(1)}$.

Since $X^\top \widetilde{p}_1(X) X, X^\top \widetilde{p}_2(X) X \in \mathbb{R}^{d \times d}$, the computation time for $X^\top (\widetilde{p}_1(X) - \widetilde{p}_2(X)) X$ is $O(d^2)$.

Therefore, the overall running time for $X^\top (\widetilde{p}_1(X) - \widetilde{p}_2(X)) X$ is $n^{1+o(1)}$.

**Proof of error bound.**

We consider the error for $X^\top \widetilde{p}_1(X) X$ first.

$$\begin{aligned} & \|X^\top \widetilde{p}_1(X) X - X^\top p_1(X) X\|_\infty \\ = & \|X^\top (\widetilde{p}_1(X) - p_1(X)) X\|_\infty \\ \leq & n^2 \|X\|_\infty^2 \|\widetilde{p}_1(X) - p_1(X)\|_\infty \\ \leq & n^2 (\epsilon / \operatorname{poly}(n)) \|X\|_\infty^2 \end{aligned}$$

$$\leq \epsilon / \mathrm{poly}(n) \tag{18}$$

where the 1st step is from basic algebra, the 2nd step comes from basic linear algebra, the 3rd step is because of $\|\widetilde{p}_1(X) - p_1(X)\|_\infty \leq \epsilon / \mathrm{poly}(n)$, the 4th step is due to $\|X\|_\infty \leq \mathrm{poly}(n)$.

Similarly, we can have

$$\|X^\top \widetilde{p}_2(X) X - X^\top p_2(X) X\|_\infty \leq \epsilon / \mathrm{poly}(n) \tag{19}$$

Therefore, we have

$$\begin{aligned}
& \|X^\top \widetilde{p}(X) X - X^\top p(X) X\|_\infty \\
= & \|X^\top \widetilde{p}_1(X) X - X^\top p_1(X) X + X^\top \widetilde{p}_2(X) X - X^\top p_2(X) X\|_\infty \\
\leq & \|X^\top \widetilde{p}_1(X) X - X^\top p_1(X) X\|_\infty + \|X^\top \widetilde{p}_2(X) X - X^\top p_2(X) X\|_\infty \\
\leq & (\epsilon / \mathrm{poly}(n)) + (\epsilon / \mathrm{poly}(n)) \\
= & \epsilon / \mathrm{poly}(n)
\end{aligned}$$

where the 1st step is from basic algebra, the 2nd step comes from triangle inequality, the 3rd step is because of Eq. (18) and Eq. (19), the 4th step is due to basic algebra.

Then, we choose $\epsilon = 1/\mathrm{poly}(n)$, we have

$$\|\widetilde{g}_w - \frac{\mathrm{d}L(X)}{\mathrm{d}W_i}\|_\infty \leq 1/\mathrm{poly}(n)$$

■

## Appendix J. Fast Computation for Gradient on $W_V$

In Section J.1, we introduce the close form of the gradient of $s(X)$ on $W_V$. In Section J.2, we provide the close form of the gradient of $L(X)$ on $W_V$. In Section J.3, based on the close form calculated in the previous section, we introduce the almost linear time algorithm for computing the gradient of $L(X)$ on $W_V$.

### J.1. Gradient of $s(X)$ on $W_V$

Since $s(X) = f(X)h(X)$, we begin with considering the gradient of $h(X)$ on $W_V$ in Lemma 67.

**Lemma 67 (Gradient of $h(X)$ on $W_V$)** *If we have the below conditions,*

- *Let $h(X)$ be defined as Definition 23.*
- *Let $W_V$ be defined as Definition 2.*

*Then, for any $i_0 \in [n], j_0 \in [d]$ and any $i_1, j_1 \in [d]$, we have*

$$\frac{\mathrm{d}h(X)_{i_0,j_0}}{\mathrm{d}(W_V)_{i_1,j_1}} = \begin{cases} X_{i_0,i_1} & j_0 = j_1 \\ 0 & j_0 \neq j_1 \end{cases}$$

**Proof** Since $h_{i_0,j_0}$ satisfies

$$h_{i_0,j_0} = X_{i_0,*}^\top (W_V)_{*,j_0},$$

we have $h_{i_0,j_0}$ only depends on $(W_V)_{*,j_0}$.

Hence, we have, for $j_0 \neq j_1$,

$$\frac{\mathrm{d}h(X)_{i_0,j_0}}{\mathrm{d}(W_V)_{i_1,j_1}} = 0$$

For $j_0 = j_1$ case, we have

$$\frac{\mathrm{d}h(X)_{i_0,j_0}}{\mathrm{d}(W_V)_{i_1,j_0}} = X_{i_0,i_1}$$

■

Combining the result in the previous Lemma and the chain rule, we can have the gradient of $s(X)$ on $W_V$ in Lemma 68.

**Lemma 68 (Gradient of $s(X)$ on $W_V$)** *If we have the below conditions,*

- *Let $s(X)$ be defined as Definition 24.*
- *Let $W_V$ be defined as Definition 2.*

*Then, for any $i_2 \in [n], j_2 \in [d]$ and any $i_1, j_1 \in [d]$, we have*

- ***Part 1.***

$$\frac{\mathrm{d}s(X)_{i_2,j_2}}{\mathrm{d}(W_V)_{i_1,j_1}} = \begin{cases} f(X)_{i_2,*}^\top X_{*,i_1} & j_2 = j_1 \\ 0 & j_2 \neq j_1 \end{cases}$$

- ***Part 2.***

$$\underbrace{\frac{\mathrm{d}s(X)_{i_2,j_2}}{\mathrm{d}W_V}}_{d\times d} = \underbrace{X^\top}_{d\times n} \underbrace{f(X)_{i_2,*}}_{n\times 1} \underbrace{e_{j_2}^\top}_{1\times d}$$

**Proof** **Proof of Part 1.**

By Definition 24, we have

$$s(X)_{i_2,j_2} := f(X)_{i_2,*}^\top h(X)_{*,j_2} \tag{20}$$

Therefore, $s(X)_{i_2,j_2}$ is only depends on $h(X)_{*,j_2}$, which further means $s(X)_{i_2,j_2}$ is only depends on $(W_V)_{*,j_2}$.

Hence, for $j_1 \neq j_2$, we have

$$\frac{\mathrm{d}s(X)_{i_2,j_2}}{\mathrm{d}(W_V)_{i_1,j_2}} = 0$$

We consider $j_1 = j_2$ case.
By, Eq. (20), we can derive that

$$\frac{\mathrm{d} s(X)_{i_2,j_2}}{\mathrm{d} h(X)_{i_3,j_2}} = f(X)_{i_2,i_3} \tag{21}$$

By chain rule, we have

$$\begin{aligned} & \frac{\mathrm{d} s(X)_{i_2,j_2}}{\mathrm{d} (W_V)_{i_1,j_2}} \\ = & \sum_{i_3=1}^{d} \frac{\mathrm{d} s(X)_{i_2,j_2}}{\mathrm{d} h(X)_{i_3,j_2}} \frac{\mathrm{d} h(X)_{i_3,j_2}}{\mathrm{d} (W_V)_{i_1,j_2}} \\ = & \sum_{i_3=1}^{d} f(X)_{i_2,i_3} \frac{\mathrm{d} h(X)_{i_3,j_2}}{\mathrm{d} (W_V)_{i_1,j_2}} \\ = & \sum_{i_3=1}^{d} f(X)_{i_2,i_3} X_{i_3,i_1} \\ = & f(X)_{i_2,*}^\top X_{*,i_1} \end{aligned} \tag{22}$$

where the 1st step is from chain rule, the 2nd step comes from Eq. (21), the 3rd step is because of Lemma 67, the 4th step is due to basic linear algebra.

**Proof of Part 2.**

By Eq (22), we have

$$\underbrace{\frac{\mathrm{d} s(X)_{i_2,j_2}}{\mathrm{d} (W_V)_{*,j_2}}}_{d\times 1} = \underbrace{X^\top}_{d\times n} \underbrace{f(X)_{i_2,*}}_{n\times 1}$$

which implies

$$\underbrace{\frac{\mathrm{d} s(X)_{i_2,j_2}}{\mathrm{d} W_V}}_{d\times d} = \underbrace{X^\top}_{d\times n} \underbrace{f(X)_{i_2,*}}_{n\times 1} \underbrace{e_{j_2}^\top}_{1\times d}$$

■

### J.2. Gradient of $L(X)$ on $W_V$

Since we have already got the close form of the gradient of $s(X)$ on $W_V$, we can easily extend it and get the close form of the gradient of $L(X)$ on $W_V$ in Lemma 69.

**Lemma 69 (Gradient of $L(X)$ on $W_V$)** *If we have the below conditions,*

- *Let $L(X)$ be defined as Definition 5.*
- *Let $W_V$ be defined as Definition 2.*

*Then, we can show that*

$$\underbrace{\frac{\mathrm{d}L(X)}{\mathrm{d}W_{V_i}}}_{d\times d} = \underbrace{X^\top}_{d\times n} \underbrace{f(X)}_{n\times n} \underbrace{G_i}_{n\times d}$$

**Proof** We slightly abuse the notation, using $W_V$ to represent $V_i$ in Lemma 67, 68.

By Lemma 68, we have

$$\underbrace{\frac{\mathrm{d}s(X)_{i_2,j_2}}{\mathrm{d}W_V}}_{d\times d} = \underbrace{X^\top}_{d\times n} \underbrace{f(X)_{i_2,*}}_{n\times 1} \underbrace{e_{j_2}^\top}_{1\times d} \tag{23}$$

By Lemma 18, we have

$$\frac{\mathrm{d}L(X)}{\mathrm{d}W_{V_i}} = \sum_{i_2=1}^{n}\sum_{j_2=1}^{d} G_i(i_2,j_2)\cdot \frac{\mathrm{d}\mathsf{Attn}_i(T_{i-1}(X))_{i_2,j_2}}{\mathrm{d}W_{V_i}}. \tag{24}$$

By Definition 24 and Definition 2, we have

$$s(X) = \mathsf{Attn}_i(T_{i-1}(X))$$

Therefore, combining Eq. (23) and Eq. (24), we have

$$\begin{aligned}
&\frac{\mathrm{d}L(X)}{\mathrm{d}W_{V_i}} \\
= & \sum_{i_2=1}^{n}\sum_{j_2=1}^{d} \underbrace{G_i(i_2,j_2)}_{1\times 1} \underbrace{X^\top}_{d\times n} \underbrace{f(X)_{i_2,*}}_{n\times 1} \underbrace{e_{j_2}^\top}_{1\times d} \\
= & \sum_{i_2=1}^{n} \underbrace{X^\top}_{d\times n} \underbrace{f(X)_{i_2,*}}_{n\times 1} \sum_{j_2=1}^{d} \underbrace{G_i(i_2,j_2)}_{1\times 1} \underbrace{e_{j_2}^\top}_{1\times d} \\
= & \sum_{i_2=1}^{n} \underbrace{X^\top}_{d\times n} \underbrace{f(X)_{i_2,*}}_{n\times 1} \underbrace{G_i(i_2,*)^\top}_{1\times d} \\
= & \underbrace{X^\top}_{d\times n} \underbrace{f(X)}_{n\times n} \underbrace{G_i}_{n\times d}
\end{aligned}$$

where the 1st step is from Eq. (23) and Eq. (24), the 2nd step comes from basic algebra, the 3rd step is because of basic linear algebra, the 4th step is due to basic linear algebra.

■

### J.3. Fast computation

Finally, we can introduce our almost linear time algorithm for computing the $L(X)$ gradient on $W_V$.

**Lemma 70 (Fast computation for $\frac{\mathrm{d}L(X)}{\mathrm{d}(W_V)_i}$)** *If we have the below conditions,*

- *Let $L(X)$ be defined as Definition 5.*
- *Let $m$ denote the number of self-attention transformer layers (see Definition 3).*
- *For any $i \in [m]$, let $W_{V_i} \in \mathbb{R}^{d \times d}$ denote the attention weight in the $i$-th transformer layer.*

*We can show that $\frac{\mathrm{d}L(X)}{\mathrm{d}W_{V_i}}$ can be approximated in $n^{1+o(1)}$ time, with $1/\operatorname{poly}(n)$ approximation error. Namely, our algorithm can output $\widetilde{g}_v$ in $n^{1+o(1)}$ time, which satisfies*

$$\|\widetilde{g}_v - \frac{\mathrm{d}L(X)}{\mathrm{d}W_{V_i}}\|_\infty \leq 1/\operatorname{poly}(n)$$

**Proof**

Recall in Lemma 27, $U_1 V_1^\top$ is the low rank approximation of $f(X)$.
Let $\widetilde{f}(X) := U_1 V_1^\top$ denote the low rank approximation of $f(X)$.
Recall in Lemma 69, we have

$$\underbrace{\frac{\mathrm{d}L(X)}{\mathrm{d}W_{V_i}}}_{d \times d} = \underbrace{X^\top}_{d \times n} \underbrace{f(X)}_{n \times n} \underbrace{G_i}_{n \times d}$$

**Proof of running time.**
We compute $X^\top \widetilde{f}(X) G_i$ in following order

- Compute $\underbrace{X^\top}_{d \times n} \cdot \underbrace{U_1}_{n \times k_1}$, which takes $n^{1+o(1)}$ time.
- Compute $\underbrace{X^\top \cdot U_1}_{d \times k_1} \cdot \underbrace{V_1^\top}_{k_1 \times n}$, which takes $n^{1+o(1)}$ time.
- Compute $\underbrace{X^\top \cdot U_1 \cdot V_1^\top}_{d \times n} \cdot \underbrace{G_i}_{n \times d}$, which takes $d^2 \cdot n$ time.

The overall running time is $n^{1+o(1)}$.
**Proof of error bound.**
We have

$$\begin{aligned}
&\|X^\top \cdot f(X) \cdot G_i - X^\top \cdot \widetilde{f}(X) \cdot G_i\|_\infty \\
= &\|X^\top \cdot (f(X) - \widetilde{f}(X)) \cdot G_i\|_\infty \\
\leq &n^2 \|X\|_\infty \|f(X) - \widetilde{f}(X)\|_\infty \|G_i\|_\infty \\
\leq &n^2 (\epsilon/\operatorname{poly}(n)) \|X\|_\infty \|G_i\|_\infty
\end{aligned}$$

$$\leq \epsilon / \mathrm{poly}(n)$$

where the 1st step is from basic algebra, the 2nd step comes from basic linear algebra, the 3rd step is because of $\|f(X) - \widetilde{f}(X)\|_\infty \leq \epsilon/\mathrm{poly}(n)$, the 4th step is due to $\|X\|_\infty \leq \mathrm{poly}(n)$ and $\|G_i\|_\infty \leq \mathrm{poly}(n)$.

Let $\widetilde{g}_v = X^\top \cdot \widetilde{f}(X) \cdot G_i$.

We choose $\epsilon = 1/\mathrm{poly}(n)$. Then, we have

$$\|\widetilde{g}_v - \frac{\mathrm{d}L(X)}{\mathrm{d}W_{V_i}}\|_\infty \leq 1/\mathrm{poly}(n)$$

■

## Appendix K. Gradient Approximation for Entire Model

In Section K.1, we introduce the close form of $G_i$ and argue that $G_i$ can be computed in almost linear time $n^{1+o(1)}$. In Section K.2, we provide the almost linear time algorithm for gradient computing on a single-layer transformer. In Section K.3, with the help of math induction, we introduce the almost linear time algorithm for computing the gradient of the multi-layer transformer, along with its approximation error.

### K.1. Computation time for $G_i$

Here we consider $g_i$ in Definition 3 as a linear layer with an arbitrary non-linear activation $\phi$. Since $g_i$ can be viewed as a composition of an MLP and an activation function, we begin with analyzing the $T_i$ gradient on $\mathsf{Attn_i}$.

**Lemma 71 (Gradient of $T_i$ on $\mathsf{Attn}_i$ )** *If we have the below conditions,*

- *Let $T_i(X)$ be defined as Definition 6.*
- *Assuming for any $Z \in \mathbb{R}^{n\times d}$, we have $g_i(Z) \in \mathbb{R}^{n\times d}$, and $g_i(Z) = \phi(ZW_g)$, where $W_g \in \mathbb{R}^{d\times d}$ and $\phi : \mathbb{R} \to \mathbb{R}$ denotes any element-wise activation function. Let $\phi'$ denote the derivative of $\phi$.*
- *We simplify the notation, using $T_i$ and $\mathsf{Attn}_i$ to represent $T_i(X)$ and $\mathsf{Attn}_i(T_{i-1}(X))$, respectively.*
- *For any matrix $Z \in \mathbb{R}^{n\times d}$, we use $Z(i,j)$ to denote the $(i,j)$-th entry of $Z$.*

*Then, we can show that, for any $i_4, i_5 \in [n], j_4, j_5 \in [d]$,*

- ***Part 1.***

$$\frac{\mathrm{d}T_i(i_4,j_4)}{\mathrm{d}\mathsf{Attn}_i(i_5,j_5)} = \begin{cases} \underbrace{\phi'(\mathsf{Attn}_i(i_4,*)^\top W_g(*,j_4))}_{1\times 1} \underbrace{W_g(j_5,j_4)}_{1\times 1} & i_4 = i_5 \\ 0 & i_4 \neq i_5 \end{cases}$$

- ***Part 2.***

$$\underbrace{\frac{\mathrm{d}T_i(i_4,j_4)}{\mathrm{d}\mathsf{Attn}_i}}_{n\times d} = \underbrace{\phi'(\mathsf{Attn}_i(i_4,*)^\top W_g(*,j_4))}_{1\times 1}\underbrace{e_{i_4}}_{n\times 1}\underbrace{W_g(*,j_4)^\top}_{1\times d}$$

**Proof** **Proof of Part 1.**

By the definition of $T_i$ (Definition 6), for $i_4 \in [d], j_4 \in [n]$, we have

$$T_i(i_4,j_4) = \phi(\mathsf{Attn}_i(i_4,*)^\top W_g(*,j_4))$$

Therefore, for any $i_5 \neq i_4$, we have

$$\frac{\mathrm{d}T_i(i_4,j_4)}{\mathrm{d}\mathsf{Attn}_i(i_5,j_5)} = 0$$

Then, we consider $i_4 = i_5$ case.

By basic calculus, we have

$$\frac{\mathrm{d}T_i(i_4,j_4)}{\mathrm{d}\mathsf{Attn}_i(i_4,j_5)} = \underbrace{\phi'(\mathsf{Attn}_i(i_4,*)^\top W_g(*,j_4))}_{1\times 1}\underbrace{W_g(j_5,j_4)}_{1\times 1}$$

Combining two equations mentioned above, we have the result for **Part 1**.

**Proof of Part 2.**

By result of **Part 1**, for $i_5 = i_4$, we have

$$\frac{\mathrm{d}T_i(i_4,j_4)}{\mathrm{d}\mathsf{Attn}_i(i_4,j_5)} = \underbrace{\phi'(\mathsf{Attn}_i(i_4,*)^\top W_g(*,j_4))}_{1\times 1}\underbrace{W_g(j_5,j_4)}_{1\times 1}$$

which implies

$$\frac{\mathrm{d}T_i(i_4,j_4)}{\mathrm{d}\mathsf{Attn}_i(i_4,*)} = \underbrace{\phi'(\mathsf{Attn}_i(i_4,*)^\top W_g(*,j_4))}_{1\times 1}\underbrace{W_g(*,j_4)}_{d\times 1}$$

By result of **Part 1**, for $i_5 \neq i_4$, we have

$$\frac{\mathrm{d}T_i(i_4,j_4)}{\mathrm{d}\mathsf{Attn}_i(i_5,*)} = 0$$

By basic linear algebra, combining the two equations mentioned above, we have

$$\frac{\mathrm{d}T_i(i_4,j_4)}{\mathrm{d}\mathsf{Attn}_i} = \underbrace{\phi'(\mathsf{Attn}_i(i_4,*)^\top W_g(*,j_4))}_{1\times 1}\underbrace{e_{i_4}}_{n\times 1}\underbrace{W_g(*,j_4)^\top}_{1\times d}$$

■

Then, we can argue that the computation for $G_i$ can be done in almost linear time $n^{1+o(1)}$.

**Lemma 72 (Computation time for $G_i$, formal version of Lemma 14)** *If we have the below conditions,*

- *Let $G_i \in \mathbb{R}^{n \times d}$ denote the gradient matrix resulting from the application of the chain rule up to the function $g_i$, i.e., $G_i = \frac{\mathrm{d}L(X)}{\mathrm{d}\mathsf{Attn}_i(T_{i-1}(X))}$.*
- *Assuming we already have $\frac{\mathrm{d}L(X)}{\mathrm{d}T_i(X)}$.*
- *Assuming for any $Z \in \mathbb{R}^{n \times d}$, we have $g_i(Z) \in \mathbb{R}^{n \times d}$, and $g_i(Z) = \phi(ZW_g)$, where $W_g \in \mathbb{R}^{d \times d}$ and $\phi : \mathbb{R} \to \mathbb{R}$ denotes any element-wise activation function. Let $\phi'$ denote the derivative of $\phi$.*
- *We simplify the notation, using $T_i$ and $\mathsf{Attn}_i$ to represent $T_i(X)$ and $\mathsf{Attn}_i(T_{i-1}(X))$, respectively.*
- *For any matrix $Z \in \mathbb{R}^{n \times d}$, we use $Z(i,j)$ to denote the $(i,j)$-th entry of $Z$.*

*Then, we can show that $G_i$ can be computed in $n^{1+o(1)}$ time.*

**Proof**

Let $g_{T_i} := \frac{\mathrm{d}L(X)}{\mathrm{d}T_i}$, and for any $i_4 \in [n], j_4 \in [d]$, let $g_{T_i}(i_4, j_4)$ denote the $(i_4, j_4)$-th entry of $g_{T_i}$.

Similarly, for any $i_5 \in [n], j_5 \in [d]$, let $T_i(i_5, j_5)$ denote the $(i_5, j_5)$-th entry of $T_i$.

We can have

$$\begin{aligned} G_i &= \frac{\mathrm{d}L(X)}{\mathrm{d}\mathsf{Attn}_i} \\ &= \frac{\mathrm{d}L(X)}{\mathrm{d}T_i} \cdot \frac{\mathrm{d}T_i}{\mathrm{d}\mathsf{Attn}_i} \\ &= g_{T_i} \cdot \frac{\mathrm{d}T_i}{\mathrm{d}\mathsf{Attn}_i} \\ &= \sum_{i_4=1}^{n} \sum_{j_4=1}^{d} g_{T_i}(i_4, j_4) \cdot \frac{\mathrm{d}T_i(i_4, j_4)}{\mathrm{d}\mathsf{Attn}_i} \end{aligned}$$

where the 1st step is from the definition of $G_i$, the 2nd step comes from chain rule, the 3rd step is because of the definition of $g_{T_i}$, the 4th step is due to chain rule.

$$\begin{aligned} & \sum_{i_4=1}^{n} \sum_{j_4=1}^{d} g_{T_i}(i_4, j_4) \cdot \frac{\mathrm{d}T_i(i_4, j_4)}{\mathrm{d}\mathsf{Attn}_i} \\ = & \sum_{i_4=1}^{n} \sum_{j_4=1}^{d} \underbrace{g_{T_i}(i_4, j_4)}_{1 \times 1} \underbrace{\phi'(\mathsf{Attn}_i(i_4, *)^\top W_g(*, j_4))}_{1 \times 1} \underbrace{e_{i_4}}_{n \times 1} \underbrace{W_g(*, j_4)^\top}_{1 \times d} \\ = & \sum_{i_4=1}^{n} \underbrace{e_{i_4}}_{n \times 1} \sum_{j_4=1}^{d} \underbrace{g_{T_i}(i_4, j_4)}_{1 \times 1} \underbrace{\phi'(\mathsf{Attn}_i(i_4, *)^\top W_g(*, j_4))}_{1 \times 1} \underbrace{W_g(*, j_4)^\top}_{1 \times d} \end{aligned}$$

$$= \sum_{i_4=1}^{n} \underbrace{e_{i_4}}_{n\times 1} ( \underbrace{W_g}_{d\times d} (\underbrace{g_{T_i}(i_4,*)}_{d\times 1} \odot \underbrace{\phi'(\mathsf{Attn}_i(i_4,*)^\top W_g)}_{d\times 1}))^\top$$
$$= \underbrace{(g_{T_i} \odot \phi'(\mathsf{Attn}_i W_g))}_{n\times d} \underbrace{W_g^\top}_{d\times d} \tag{25}$$

where the 1st step is from Lemma 71, the 2nd step comes from basic algebra, the 3rd step is because of basic linear algebra, the 4th step is due to basic linear algebra.

By Eq. (25), we have the close form of $G_i$.

We can compute $G_i$ in the following order

- Compute $\underbrace{(g_{T_i} \odot \phi'(\mathsf{Attn}_i W_g))}_{n\times d}$, which takes $n \cdot d$ time.
- Compute $\underbrace{(g_{T_i} \odot \phi'(\mathsf{Attn}_i W_g))}_{n\times d} \underbrace{W_g^\top}_{d\times d}$, which takes $d^2 \cdot n$ time.

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

■

### K.2. Fast computation for single-layer transformer

In this section, we dive into the computation time and approximation error of the gradient of a single-layer transformer. We demonstrate in the following Lemma that the gradient of a single-layer transformer can be computed in almost linear time $n^{1+o(1)}$, and its error can be bounded by $1/\operatorname{poly}(n)$.

**Lemma 73 (Single-layer transformer gradient approximation)** *If we have the below conditions,*

- *Let $L(X)$ be defined as Definition 5.*
- *Let $X$ be defined as Definition 2.*
- *Let the gradient matrix $G_i \in \mathbb{R}^{n\times d}$ be defined as $G_i = \frac{\mathrm{d}L(X)}{\mathrm{d}\mathsf{Attn}_i(T_{i-1}(X))}$.*
- *For $i_2 \in [n], j_2 \in [d]$, let $G_i(i_2, j_2)$ denote the $(i_2, j_2)$-th entry of $G_i$.*
- *Assuming for any $Z \in \mathbb{R}^{n\times d}$, we have $g_i(Z) \in \mathbb{R}^{n\times d}$, and $g_i(Z) = \phi(Z \cdot W_g)$, where $W_g \in \mathbb{R}^{d\times d}$ and $\phi : \mathbb{R} \to \mathbb{R}$ denotes any element-wise activation function. Let $\phi'$ denote the derivative of $\phi$.*
- *Suppose we have a single-layer transformer (see Definition 3).*

*Then, we can show that,*

- ***Part 1: running time.** Our algorithm can approximate $\frac{\mathrm{d}L(X)}{\mathrm{d}X}$ in $n^{1+o(1)}$ time.*

- ***Part 2: error bound.*** *The approximation error of the single-layer transformer can be bounded by* $1/\operatorname{poly}(n)$*. Namely, our algorithm output* $\widetilde{g}_1$ *satisfies*

$$\|\widetilde{g}_1 - \frac{\mathrm{d}L(X)}{\mathrm{d}X}\|_\infty \leq 1/\operatorname{poly}(n)$$

**Proof** By Definition 3, a single-layer transformer has following structure:

$$g_1 \circ \mathsf{Attn}_1 \circ g_0(X)$$

By the definition of $G_i$, we have

$$\begin{aligned} G_1 &= \frac{\mathrm{d}L(X)}{\mathrm{d}\mathsf{Attn}_1(T_0(X))} \\ &= \frac{\mathrm{d}L(X)}{\mathrm{d}T_1(X)} \cdot \frac{\mathrm{d}T_1(X)}{\mathrm{d}\mathsf{Attn}_1(T_0(X))} \end{aligned} \tag{26}$$

By Lemma 72, we have $G_1$ can be computed in $n^{1+o(1)}$ time.

**Proof of Part 1: running time.**

For less confusion, in this part of the proof, we ignore the approximation error temporarily.

Since we have got $G_1$, we use methods mentioned in Lemma 61, 66, 70 to compute $\frac{\mathrm{d}L(X)}{\mathrm{d}T_0(X)}$, $\frac{\mathrm{d}L(X)}{\mathrm{d}W_1}$, $\frac{\mathrm{d}L(X)}{\mathrm{d}W_{V_1}}$, respectively, which takes $n^{1+o(1)}$ time for each.

Then, since we have $\frac{\mathrm{d}L(X)}{\mathrm{d}T_0(X)}$, again by Lemma 72, we have $\frac{\mathrm{d}L(X)}{\mathrm{d}X}$ can be computed in $n^{1+o(1)}$ time.

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

**Proof of Part 2: error bound.**

Then, we move on to the error bound.

By Lemma 72 and Eq. (26), there is no approximation error when computing $G_1$.

By Lemma 61, 66, 70, we have there is $1/\operatorname{poly}(n)$ approximation error on $\frac{\mathrm{d}L(X)}{\mathrm{d}T_0(X)}$, $\frac{\mathrm{d}L(X)}{\mathrm{d}W_1}$, $\frac{\mathrm{d}L(X)}{\mathrm{d}W_{V_1}}$, respectively.

Let $\widetilde{g}_{t_0}, \widetilde{g}_{w_1}, \widetilde{g}_{v_1}$ denote the approximation results of $\frac{\mathrm{d}L(X)}{\mathrm{d}T_0(X)}$, $\frac{\mathrm{d}L(X)}{\mathrm{d}W_1}$, $\frac{\mathrm{d}L(X)}{\mathrm{d}W_{V_1}}$, respectively.

We have

$$\|\widetilde{g}_{t_0} - \frac{\mathrm{d}L(X)}{\mathrm{d}T_0(X)}\|_\infty \leq 1/\operatorname{poly}(n) \tag{27}$$

and

$$\|\widetilde{g}_{w_1} - \frac{\mathrm{d}L(X)}{\mathrm{d}W_1}\|_\infty \leq 1/\operatorname{poly}(n)$$

and

$$\|\widetilde{g}_{v_1} - \frac{\mathrm{d}L(X)}{\mathrm{d}W_{V_1}}\|_\infty \leq 1/\operatorname{poly}(n)$$

Let $\widetilde{G}_0 = \widetilde{g}_{t_0} \cdot \frac{\mathrm{d}T_0(X)}{\mathrm{d}X}$ denote the approximated version of $G_0$.

We have

$$\begin{aligned}
& \|\widetilde{G}_0 - G_0\|_\infty \\
= & \|(\widetilde{g}_{t_0} - \frac{\mathrm{d}L(X)}{\mathrm{d}T_0(X)}) \cdot \frac{\mathrm{d}T_0(X)}{\mathrm{d}X}\|_\infty \\
\leq & n \cdot d\|\widetilde{g}_{t_0} - \frac{\mathrm{d}L(X)}{\mathrm{d}T_0(X)}\|_\infty \|\frac{\mathrm{d}T_0(X)}{\mathrm{d}X}\|_\infty \\
\leq & n \cdot d(1/\operatorname{poly}(n))\|\frac{\mathrm{d}T_0(X)}{\mathrm{d}X}\|_\infty \\
\leq & 1/\operatorname{poly}(n)
\end{aligned}$$

where the 1st step is from the definition of $\widetilde{G}_0$, the 2nd step comes from basic linear algebra, the 3rd step is because of Eq. (27), the 4th step is due to each entry can be written by $O(\log n)$ bits.

Let $\widetilde{g}_1 = \widetilde{G}_0$.

Therefore, we have

$$\|\widetilde{g}_1 - \frac{\mathrm{d}L(X)}{\mathrm{d}X}\|_\infty \leq 1/\operatorname{poly}(n)$$

■

### K.3. Fast computation for multi-layer transformer

Since we have already demonstrated that almost linear time gradient computation can be applied to a single-layer transformer, with the help of math induction, we can easily generalize that result to the multi-layer transformer. In the following Lemma, we display that the gradient of the multi-layer transformer can be computed in almost linear time, and its approximation error can be bounded by $1/\operatorname{poly}(n)$.

**Lemma 74 (Multi-layer transformer gradient approximation, formal version of Lemma 15)** *If we have the below conditions,*

- *Let $L(X)$ be defined as Definition 5.*
- *Let $X$ be defined as Definition 2.*
- *Let $G_i \in \mathbb{R}^{n \times d}$ denote the gradient matrix resulting from the application of the chain rule up to the function $g_i$, i.e., $G_i = \frac{\mathrm{d}L(X)}{\mathrm{d}\mathsf{Attn}_i(T_{i-1}(X))}$.*
- *For $i_2 \in [n], j_2 \in [d]$, let $G_i(i_2, j_2)$ denote the $(i_2, j_2)$-th entry of $G_i$.*
- *Let gradient components for each layer be computed according to Lemma 61, 66, 70.*
- *Assuming for any $Z \in \mathbb{R}^{n \times d}$, we have $g_i(Z) \in \mathbb{R}^{n \times d}$, and $g_i(Z) = \phi(Z \cdot W_g)$, where $W_g \in \mathbb{R}^{d \times d}$ and $\phi : \mathbb{R} \to \mathbb{R}$ denotes any element-wise activation function. Let $\phi'$ denote the derivative of $\phi$.*
- *Suppose we have a $m$-layer transformer (see Definition 3).*

*Then, we can show that,*

- ***Part 1: running time.*** *Our algorithm can approximate* $\frac{\mathrm{d}L(X)}{\mathrm{d}X}$ *in* $n^{1+o(1)}$ *time.*
- ***Part 2: error bound.*** *The approximation error of the multi-layer transformer can be bounded by* $1/\operatorname{poly}(n)$*. Namely, our algorithm output* $\widetilde{g}$ *satisfies*

$$\|\widetilde{g} - \frac{\mathrm{d}L(X)}{\mathrm{d}X}\|_\infty \leq 1/\operatorname{poly}(n)$$

**Proof** We use math induction to prove this Lemma.

**Step 1: Proof of a single-layer transformer.**

Firstly, by Lemma 73, we have that for one-layer transformer, our conclusion is established.

**Step 2: Assumption for $k$-layer transformer.**

Secondly, we assume for any $k$, for $k$-layer transformer model, we have

- Our algorithm can approximate $\frac{\mathrm{d}L(X)}{\mathrm{d}X}$ in $O(n^{1+o(1)})$ time.
- The approximation error of the $k$-layer transformer can be bounded by $1/\operatorname{poly}(n)$. Namely, our algorithm output $\widetilde{g}$ satisfies

$$\|\widetilde{g} - \frac{\mathrm{d}L(X)}{\mathrm{d}X}\|_\infty \leq 1/\operatorname{poly}(n)$$

**Step 3: Proof of $(k+1)$-layer transformer.**

Thirdly, we consider the $(k+1)$-layer transformer model.

Without loss of generality, we assume that the additional transformer layer is added at the beginning of the model.

Namely, let $\mathsf{F}_k$ denote a $k$-layer transformer model. We have

$$\mathsf{F}_k(X) = g_k \circ \mathsf{Attn}_k \circ \cdots \circ g_1 \circ \mathsf{Attn}_1 \circ g_0(X)$$

Let the $(k+1)$-layer transformer model have the following structure:

$$\mathsf{F}_{k+1}(X) = \mathsf{F}_k \circ \mathsf{Attn} \circ g(X) \tag{28}$$

Let $T_0 := g(X)$.

By assumption, we have

- $\frac{\mathrm{d}L(X)}{\mathrm{d}\mathsf{Attn}(T_0)}$ can be approximated in $n^{1+o(1)}$ time.
- Let $\widetilde{g}_k$ denote the approximated version of $\frac{\mathrm{d}L(X)}{\mathrm{d}\mathsf{Attn}(T_0)}$. We have

$$\|\widetilde{g}_k - \frac{\mathrm{d}L(X)}{\mathrm{d}\mathsf{Attn}(T_0)}\|_\infty \leq 1/\operatorname{poly}(n) \tag{29}$$

**Step 3.1: Proof of the running time for $(k+1)$-layer transformer**

For less confusion, in this part of the proof, we ignore the approximation error temporarily.

By the assumption, we have $\frac{\mathrm{d}L(X)}{\mathrm{d}\mathsf{Attn}(T_0)}$ can be approximated in $n^{1+o(1)}$ time.

We compute $\frac{\mathrm{d}L(X)}{\mathrm{d}X}$ in following order:

- Since we already have $\frac{\mathrm{d}L(X)}{\mathrm{d}\mathsf{Attn}(T_0)}$, by Lemma 61, the computation time for $\frac{\mathrm{d}L(X)}{\mathrm{d}T_0}$ is $n^{1+o(1)}$.
- Since we have $\frac{\mathrm{d}L(X)}{\mathrm{d}T_0}$, by Lemma 72, the computation time for $\frac{\mathrm{d}L(X)}{\mathrm{d}X}$ is $n^{1+o(1)}$.

Therefore, for $(k+1)$-layer transformer, the overall running time for $\frac{\mathrm{d}L(X)}{\mathrm{d}X}$ is $n^{1+o(1)}$.

**Step 3.2: Proof of the error bound for $(k+1)$-layer transformer**

By Lemma 61, during the process of solving the approximated version of $\frac{\mathrm{d}L(X)}{\mathrm{d}g(X)}$, the approximation error will not be magnified by more than $\mathrm{poly}(n)$.

Let $\widetilde{g}_{t_0}$ denote the approximated version of $\frac{\mathrm{d}L(X)}{\mathrm{d}g(X)}$, we have

$$\begin{aligned}
& \|\widetilde{g}_{t_0} - \frac{\mathrm{d}L(X)}{\mathrm{d}g(X)}\|_\infty \\
\leq & \ \mathrm{poly}(n)\|\widetilde{g}_k - \frac{\mathrm{d}L(X)}{\mathrm{d}T(X)}\|_\infty \\
\leq & \ 1/\mathrm{poly}(n)
\end{aligned} \tag{30}$$

where the 1st step is from the above statement, the 2nd step comes from Eq. (29), the 3rd step is because of basic algebra.

Then, we consider

$$\frac{\mathrm{d}L(X)}{\mathrm{d}X} = \frac{\mathrm{d}L(X)}{\mathrm{d}g(X)} \cdot \frac{\mathrm{d}g(X)}{\mathrm{d}X} \tag{31}$$

Recall that we have $\widetilde{g} = \frac{\mathrm{d}L(X)}{\mathrm{d}X}$. Then, we have

$$\begin{aligned}
& \|\widetilde{g} - \frac{\mathrm{d}L(X)}{\mathrm{d}X}\|_\infty \\
= & \ \|(\widetilde{g}_{t_0} - \frac{\mathrm{d}L(X)}{\mathrm{d}g(X)}) \cdot \frac{\mathrm{d}g(X)}{\mathrm{d}X}\|_\infty \\
\leq & \ n \cdot d\|\widetilde{g}_{t_0} - \frac{\mathrm{d}L(X)}{\mathrm{d}g(X)}\|_\infty \|\frac{\mathrm{d}g(X)}{\mathrm{d}X}\|_\infty \\
\leq & \ n \cdot d(1/\mathrm{poly}(n))\|\frac{\mathrm{d}g(X)}{\mathrm{d}X}\|_\infty \\
\leq & \ 1/\mathrm{poly}(n)
\end{aligned}$$

where the 1st step is from Eq. (31), the 2nd step comes from basic linear algebra, the 3rd step is because of Eq. (30), the 4th step is due to each entry can be written by $O(\log n)$ bits.

**Step 4: Use math induction.**

So far, with the assumption that our statement holds under $k$-layer transformer, we have proved that our statement still holds under $(k+1)$-layer transformer.

Therefore, by math induction, our statement holds for any $m$-layer transformer.

■

## Appendix L. Causal Attention Mask

This section will discuss how to combine the causal attention mask with our framework. We argue that even with the causal attention mask, we can also achieve almost linear time gradient computing for the multi-layer transformer.

In Section L.1, we introduce essential tools from literature to deal with the causal mask added on the attention matrix. In Section L.2, we show that with the addition of causal mask, our framework can still achieve almost linear time gradient computation.

### L.1. Tools from previous work

Firstly, we restate a classical low-rank approximation method in the literature.

**Lemma 75 (Low-rank approximation, [3])** *Suppose $Q, K \in \mathbb{R}^{n \times d}$, with $\|Q\|_\infty \leq R$, and $\|K\|_\infty \leq R$. Let $A := \exp(QK^\top/d) \in \mathbb{R}^{n \times n}$. For accuracy parameter $\epsilon \in (0,1)$, there is a positive integer $g$ bounded above by*

$$g = O\Big( \max \Big\{ \frac{\log(1/\epsilon)}{\log(\log(1/\epsilon)/R)}, R^2 \Big\} \Big),$$

*and a positive integer $r$ bounded above by*

$$r \leq \binom{2(g+d)}{2g}$$

*such that: There is a matrix $\widetilde{A} \in \mathbb{R}^{n \times n}$ that is an $(\epsilon, r)$-approximation of $A \in \mathbb{R}^{n \times n}$. Furthermore, the matrices $U_0$ and $V_0$ defining $\widetilde{A}$ can be computed in $O(n \cdot r)$ time.*

Then, we provide the formal definition for the causal attention mask.

**Definition 76 (Causal attention mask, [72])** *We define the causal attention mask as $M \in \{0,1\}^{n \times n}$, where $M_{i,j} = 1$ if $i \geq j$ and $M_{i,j} = 0$ otherwise.*

In previous work [72], they point out there exists an algorithm (Algorithm 2) that can calculate low-rank matrices (with the causal attention mask) multiplication with any vector $v$ in almost linear time. We restate their results in Lemma 77.

**Lemma 77 (Fast computation for causal attention mask on tensor, [72])** *Let $M \in \{0,1\}^{n \times n}$ be a causal attention mask defined in Definition 76. Let $U_0, V_0 \in \mathbb{R}^{n \times k}$. Let $v \in \mathbb{R}^n$. Then, there exists an algorithm (see Algorithm 2) whose output satisfies that*

$$Y = (M \odot (U_0 V_0^\top))v,$$

*which takes $O(nk)$ time.*

We extend their results to the multiplication of matrix with $n^{o(1)}$ columns.

**Lemma 78 (Fast computation for causal attention mask on matrix)** *If we have the below conditions,*

**Algorithm 2** Causal attention mask algorithm, Algorithm 4 in Liang et al. [72]

**procedure** CAUSALMASK($U_0 \in \mathbb{R}^{n \times k}, V_0 \in \mathbb{R}^{n \times k}, v \in \mathbb{R}^n$) ▷ Lemma 77
  $c_0 \leftarrow \mathbf{0}_k$
  **for** $j = 1 \to n$ **do**
    $b_j \leftarrow \underbrace{(V_0^\top)_j}_{k \times 1} \underbrace{v_j}_{\mathrm{scalar}}$ ▷ Let $(V_0^\top)_j$ denote the $j$-th row of $V_0 \in \mathbb{R}^{n \times k}$
    $c_j \leftarrow \underbrace{c_{j-1}}_{k \times 1} + \underbrace{b_j}_{k \times 1}$
  **end for**
  **for** $j = 1 \to n$ **do**
    $Y_j \leftarrow \langle \underbrace{(U_0^\top)_j}_{k \times 1}, \underbrace{c_j}_{k \times 1} \rangle$
  **end for**
  **return** $Y$ ▷ $Y \in \mathbb{R}^n$
**end procedure**

- *Let $M \in \{0,1\}^{n \times n}$ be a causal attention mask defined in Definition 76.*
- *Let $U_0, V_0 \in \mathbb{R}^{n \times k}$ where $k = n^{o(1)}$.*
- *Let $H \in \mathbb{R}^{n \times k_H}$ where $k_H = n^{o(1)}$.*

*Then, there exists an algorithm, whose output satisfies that*

$$Z = (M \odot (U_0 V_0^\top)) H,$$

*which takes $n^{1+o(1)}$ time.*

**Proof** For $j \in [k_H]$, let $H_{*,j} \in \mathbb{R}^n$ denote the $j$-th column of $H$.

By Lemma 77, we can compute $(M \odot (U_0 V_0^\top)) H_{*,j}$ in $O(nk)$ time.

There are $k_H$ columns in total. Therefore, the overall running time is $O(nkk_H) = O(n \cdot n^{o(1)} \cdot n^{o(1)}) = n^{1+o(1)}$. ■

### L.2. Fast computation with causal mask

We can easily change all low-rank matrices multiplication to the algorithm mentioned in Lemma 78. Then, our framework can support the causal attention mask and still achieves almost linear time gradient computing for the multi-layer transformer.

The causal mask directly affects the attention matrix, so it's necessary to define the attention matrix with the causal mask applied.

**Definition 79** *Let $M \in \{0,1\}^{n \times n}$ be a causal attention mask defined in Definition 76. We define attention matrix with causal mask as:*

$$\widehat{f}(X) := D^{-1}(M \odot A)$$

*where $A := \exp(XWX^\top/d)$ and $D := \mathrm{diag}((M \odot A) \cdot \mathbf{1}_n)$.*

After analyzing the components of gradients on $T_i(X), W_i, W_{V_i}$ in Section H, I and J, we categorize them into two groups: one involving the dot product and the other involving the Hadamard product of the attention matrix. Then, we can show $\widehat{f}(X)H$ and $(\widehat{f}(X) \odot (UV^\top))H$ for low rank matrices $U, V, H$ can be approximated in almost linear time.

**Lemma 80** *If we have the below conditions,*

- *Let $\widehat{f}(X)$ be defined in Definition 79.*
- *Let $U, V \in \mathbb{R}^{n \times k}$ where $k = n^{o(1)}$.*
- *Let $H \in \mathbb{R}^{n \times k_H}$ where $k_H = n^{o(1)}$.*

*Then, approximating the following takes $n^{1+o(1)}$ time:*

- *Part 1. $\widehat{f}(X)H$*
- *Part 2. $(\widehat{f}(X) \odot (UV^\top))H$*

**Proof** From Definition 79, we know

$$\widehat{f}(X) := D^{-1}(M \odot A)$$

where $D := \mathrm{diag}((M \odot A) \cdot \mathbf{1}_n)$.

By Lemma 75, $U_0 V_0^\top$ is a good approximation for $A$. Then, we can approximate $\widehat{f}(X)$ by:

$$D^{-1}(M \odot (U_0 V_0^\top))$$

where $D := \mathrm{diag}((M \odot (U_0 V_0^\top)) \cdot \mathbf{1}_n)$.

Using Lemma 77, we know $(M \odot (U_0 V_0^\top)) \cdot v$ for any vector $v \in \mathbb{R}^n$ can be computed in almost linear time.

We begin by examining the normalization matrix $D^{-1}$. Calling Lemma 77, we compute $(M \odot (U_0 V_0^\top)) \cdot \mathbf{1}_n$ in almost linear time. Then, it takes $O(n)$ time to make $(M \odot (U_0 V_0^\top)) \cdot \mathbf{1}_n$ diagonal. Given that $D$ is diagonal, its inverse $D^{-1}$ can be determined in $O(n)$ time. Thus, we can compute $D^{-1}$ in almost linear time.

**Proof of Part 1.** $H$ can be viewed as a combination of $k_H$ vectors, each of size $n$. Calling Lemma 78, we can compute $(M \odot (U_0 V_0^\top))H$ in $n^{1+o(1)}$ time.

Finally, we compute $\underbrace{D^{-1}}_{n \times n} \underbrace{(M \odot (U_0 V_0^\top))H}_{n \times k_H}$, which takes $n^{1+o(1)}$ time since $D^{-1}$ is diagonal.
The overall gradient computation remains $n^{1+o(1)}$ time.

**Proof of Part 2.** The proof for this part involves Fact 17. We can show

$$\begin{aligned}
&((D^{-1}(M \odot (U_0 V_0^\top))) \odot (UV^\top))H \\
= &((M \odot (D^{-1} U_0 V_0^\top)) \odot (UV^\top))H \\
= &(M \odot ((D^{-1} U_0 V_0^\top) \odot (UV^\top)))H \\
= &(M \odot ((D^{-1} U_0) \oslash U)(V_0 \oslash V)^\top)H
\end{aligned}$$

where the 1st step is from $D(A \odot B) = (DA) \odot B = A \odot (DB)$ for diagonal matrix $D \in \mathbb{R}^{m \times m}$ and $A, B \in \mathbb{R}^{m \times n}$, the 2nd step comes from $(A \odot B) \odot C = A \odot (B \odot C)$ for $A, B, C \in \mathbb{R}^{m \times n}$, and the last step follows from Fact 17.

Let $U_M := (D^{-1} U_0) \oslash U$ and $V_M := V_0 \oslash V$.

For $U_M$, we compute $\underbrace{D^{-1}}_{n \times n} \underbrace{U_0}_{n \times k}$ which takes $nk$ time. We then compute $\underbrace{(D^{-1} U_0)}_{n \times k} \oslash \underbrace{U}_{n \times k}$ which takes $O(nk^2)$ time.

For $V_M$, we compute $\underbrace{V_0}_{n \times k} \oslash \underbrace{V}_{n \times k}$ which takes $O(nk^2)$ time.

We now have $(M \odot (U_M V_M^\top) H$. Calling Lemma 78, we finish the proof. ■

We now prove for gradient components that have dot product.

**Lemma 81 (Components for dot product)** *If we have the below conditions,*

- *Let $\widehat{f}(X)$ be defined in Definition 79.*
- *Let $G_i \in \mathbb{R}^{n \times d}$ denote the gradient matrix resulting from the application of the chain rule up to the function $g_i$, i.e., $G_i = \frac{\mathrm{d} L(X)}{\mathrm{d} \mathsf{Attn}_i (T_{i-1}(X))}$.*
- *Let $D_6 = -f(X) \operatorname{diag}(K) X W^\top$ be defined in Lemma 50.*
- *Let $D_2 = -\operatorname{diag}(K) f(X) X W$ be defined in Lemma 50.*
- *Let $D_8 = f(X) G_i W_V^\top$ be defined in Lemma 50.*
- *Let $g_v := X^\top f(X) G_i$ be the gradient on $W_{V_i}$ and defined in Lemma 69.*

*Then, we can show the following can be approximated in almost linear time:*

- *Part 1.* $\widehat{D}_6 = -\widehat{f}(X) \operatorname{diag}(K) X W^\top$
- *Part 2.* $\widehat{D}_2 = -\operatorname{diag}(K) \widehat{f}(X) X W$
- *Part 3.* $\widehat{D}_8 = \widehat{f}(X) G_i W_V^\top$
- *Part 4.* $\widehat{g}_v := X^\top \widehat{f}(X) G_i$

**Proof** **Proof of Part 1.** For $\widehat{D}_6$, we compute $\underbrace{\operatorname{diag}(K)}_{n \times n} \underbrace{X}_{n \times d}$ first, which takes $nd$ time.

Then, we compute $\underbrace{\widehat{f}(X)}_{n \times n} \underbrace{\operatorname{diag}(K) X}_{n \times d}$ using **Part 1.** of Lemma 80, which takes $n^{1+o(1)}$ time.

Finally, we compute $\underbrace{\widehat{f}(X) \operatorname{diag}(K) X}_{n \times d} \underbrace{W^\top}_{d \times d}$, which takes $n^{1+o(1)}$ time.

**Proof of Part 2.** For $\widehat{D}_2$, we compute $\underbrace{\widehat{f}(X)}_{n \times n} \underbrace{X}_{n \times d}$ using **Part 1.** of Lemma 80, which takes $n^{1+o(1)}$ time.

Then, we compute $\underbrace{\operatorname{diag}(K)}_{n \times n} \underbrace{\widehat{f}(X) X}_{n \times d}$, which takes $nd$ time.

After that, we compute $\underbrace{\mathrm{diag}(K)\widehat{f}(X)X}_{n\times d}\underbrace{W}_{d\times d}$, which takes $n^{1+o(1)}$ time.

**Proof of Part 3.** For $\widehat{D}_8$, we compute in the following steps:
We compute $\underbrace{\widehat{f}(X)}_{n\times n}\underbrace{G_i}_{n\times d}$ using **Part 1.** of Lemma 80, which takes $n^{1+o(1)}$ time.
Then, we compute $\underbrace{\widehat{f}(X)G_i}_{n\times d}\underbrace{W_V^\top}_{d\times d}$, which takes $n \cdot d^2$ time.

**Proof of Part 4.** For $\widehat{g}_v$, we compute in the following steps:
We compute $\underbrace{\widehat{f}(X)}_{n\times n}\underbrace{G_i}_{n\times d}$ using **Part 1.** of Lemma 80, which takes $n^{1+o(1)}$ time.
Then, we compute $\underbrace{X^\top}_{d\times n}\underbrace{\widehat{f}(X)G_i}_{n\times d}$, which takes $n \cdot d^2$ time. ■

We then prove for gradient components that have Hadamard product.

**Lemma 82 (Components for Hadamard product)** *If we have the below conditions,*

- *Let $\widehat{f}(X)$ be defined in Definition 79.*
- *Let $G_i \in \mathbb{R}^{n\times d}$ denote the gradient matrix resulting from the application of the chain rule up to the function $g_i$, i.e., $G_i = \frac{\mathrm{d}L(X)}{\mathrm{dAttn}_i(T_{i-1}(X))}$.*
- *Let $D_7 = (f(X) \odot (h(X)G_i^\top))XW^\top$ be defined in Lemma 50.*
- *Let $D_4 = (f(X) \odot (G_i h(X)^\top))XW$ be defined in Lemma 50.*
- *Let $g_w := X^\top p(X)X = X^\top(p_1(X) - p_2(X))X$ be the gradient on $W_i$ and defined in Definition 26 and Lemma 66 where $p_1(X) = f(X) \odot q(X)$ and $p_2(X) = \mathrm{diag}(p_1(X) \cdot 1_n)f(X)$.*

*Then, we can show the following can be approximated in almost linear time:*

- *Part 1. $\widehat{D}_7 = (\widehat{f}(X) \odot (h(X)G_i^\top))XW^\top$*
- *Part 2. $\widehat{D}_4 = (\widehat{f}(X) \odot (G_i h(X)^\top))XW$*
- *Part 3. $\widehat{g}_w := X^\top(\widehat{p}_1(X) - \widehat{p}_2(X))X$ where $\widehat{p}_1(X) = \widehat{f}(X) \odot q(X)$ and $p_2(X) = \mathrm{diag}(\widehat{p}_1(X) \cdot 1_n)\widehat{f}(X)$.*

**Proof** **Proof of Part 1.** For $\widehat{D}_7$, we can compute $\underbrace{(\widehat{f}(X) \odot (h(X)G_i^\top))}_{n\times n}\underbrace{X}_{n\times d}$ using **Part 2.** of Lemma 80, which takes $n^{1+o(1)}$ time.

We then compute $\underbrace{(\widehat{f}(X) \odot (h(X)G_i^\top))X}_{n\times d}\underbrace{W^\top}_{d\times d}$, which takes $nd^2$ time.

**Proof of Part 2.** For $\widehat{D}_7$, we can compute $\underbrace{(\widehat{f}(X) \odot (G_i h(X)^\top))}_{n\times n}\underbrace{X}_{n\times d}$ using **Part 2.** of Lemma 80, which takes $n^{1+o(1)}$ time.

We then compute $\underbrace{(\widehat{f}(X) \odot (G_i h(X)^\top))X}_{n \times d} \underbrace{W}_{d \times d}$, which takes $nd^2$ time.

**Proof of Part 3.** For $\widehat{g}_w$, we consider $X^\top \widehat{p}_1(X)X$ first. Based on Definition 25, we have $\widehat{p}_1(X) = \widehat{f}(X) \odot q(X) = \widehat{f}(X) \odot (G_i h(X)^\top)$. We then compute $(\widehat{f}(X) \odot (G_i h(X)^\top))X$ using **Part 2.** of Lemma 80, which takes $n^{1+o(1)}$ time. After that, we compute $\underbrace{X^\top}_{d \times n} \underbrace{(\widehat{f}(X) \odot (G_i h(X)^\top))X}_{n \times d}$, which takes $nd^2$ time.

Now we consider $X^\top \widehat{p}_2(X)X$. By definition, $\widehat{p}_2(X) = \mathrm{diag}(\widehat{p}_1(X) \cdot 1_n)\widehat{f}(X)$. We first compute $\widehat{p}_1(X) \cdot 1_n = (\widehat{f}(X) \odot (G_i h(X)^\top)) \cdot 1_n$ using **Part 2.** of Lemma 80, which takes $n^{1+o(1)}$ time. Meanwhile, we compute $\widehat{f}(X)X$ using **Part 1.** of Lemma 80, which takes $n^{1+o(1)}$ time. We then have $\underbrace{\mathrm{diag}(\widehat{p}_1(X) \cdot 1_n)}_{n \times n} \underbrace{\widehat{f}(X)X}_{n \times d}$, which takes $nd$ time. Finally, we compute $\underbrace{X^\top}_{d \times n} \underbrace{\mathrm{diag}(\widehat{p}_1(X) \cdot 1_n)\widehat{f}(X)X}_{n \times d}$, which takes $nd^2$ time.

Together, $\underbrace{X^\top \widehat{p}_1(X)X}_{d \times d} - \underbrace{X^\top \widehat{p}_2(X)X}_{d \times d}$ takes $d^2$ time. ■

Thus, we show that our framework can support causal attention masks.

## Appendix M. Residual Connection

In this section, we discuss how to adapt our framework to the attention mechanism with the residual connection.

In Section M.1, we provide a formalized definition of the two residual connections used in the attention mechanism. In Section M.2, we argue that with the addition of the residual connection, the gradient over the attention mechanism can be computed in almost linear time $n^{1+o(1)}$ and the approximation error can be bound by $1/\mathrm{poly}(n)$. In Section M.3, we use math induction to show that the gradient over the entire transformer with the residual connection can also be computed in almost linear time $n^{1+o(1)}$.

### M.1. Key concepts

Recall that in Definition 6, we have defined $T_i(X) \in \mathbb{R}^{n \times d}$ as the intermediate variable output by the $i$-th transformer layer. For simplicity, we use $T_i$ to represent $T_i(X)$ in the rest part of this section. Namely, we have

$$T_i = (g_i \circ \mathsf{Attn}_i)(T_{i-1})$$

Then, we consider adding the residual connection to our framework. Note that there are two residual connection operations in one transformer layer. We first define the residual connection over the $\mathsf{Attn}_i$ in Definition 83.

**Definition 83 (Residual connection over $\mathsf{Attn}_i$)** *If we have the below conditions,*

- *Let $T_i$ be defined as Definition 6.*
- *Let $\mathsf{Attn}_i$ be defined as Definition 2.*

*We define $Z_i \in \mathbb{R}^{n \times d}$ as the output with the residual connection of $\mathsf{Attn}_i$. Namely, we have*

$$Z_i = T_{i-1} + \mathsf{Attn}_i(T_{i-1})$$

Then, we consider the second residual connection over the MLP layer $g_i$, where we have the formal definition for this in Definition 84.

**Definition 84 (Residual connection over $g_i$)** *If we have the below conditions,*

- *Let the multi-layer transformer be defined as Definition 3.*
- *Let the intermediate variable $T_i$ be defined as Definition 6.*
- *Let $g_i$ denote the components other than self-attention in the $i$-th transformer layer.*
- *Let $Z_i \in \mathbb{R}^{n \times d}$ be defined as Definition 83.*

*Then $T_i$, the output of $i$-th layer transformer with the residual connection, should have the following form:*

$$T_i = Z_i + g_i(Z_i)$$

## M.2. Analysis of the residual connection

In the previous section, we have defined the two residual connection operations.

In this section, we argue that if the gradient computation can be done in almost linear time without the residual connection, then with the addition of the residual connection, the gradient computation can also be completed in almost linear time.

**Lemma 85 (Analysis of the residual connection)** *If we have the below conditions,*

- *Let $L(X)$ be defined as Definition 5.*
- *Let $Y_R \in \mathbb{R}^{n \times d}$ and $X_R \in \mathbb{R}^{n \times d}$ denote the output and input of the residual connection, respectively.*
- *Let $\mathsf{H} : \mathbb{R}^{n \times d} \to \mathbb{R}^{n \times d}$ denote some layer in the transformer, such as MLP, $\mathsf{Attn}$, etc.*
- *Suppose the residual connection can be written as*

$$Y_R = X_R + \mathsf{H}(X_R).$$

- *Assuming we have $\frac{\mathrm{d}L(X)}{\mathrm{d}Y_R} \in \mathbb{R}^{n \times d}$, then we can calculate $\frac{\mathrm{d}L(X)}{\mathrm{d}Y_R} \frac{\mathrm{d}\mathsf{H}(X_R)}{\mathrm{d}X_R}$ in almost linear time $n^{1+o(1)}$.*

*Then, we can show that,*

- *$\frac{\mathrm{d}L(X)}{\mathrm{d}X_R}$ can be calculated in almost linear time $n^{1+o(1)}$.*
- *If $\frac{\mathrm{d}L(X)}{\mathrm{d}Y_R}$ has $1/\operatorname{poly}(n)$ approximation error, then the approximation error on $\frac{\mathrm{d}L(X)}{\mathrm{d}X_R}$ is still $1/\operatorname{poly}(n)$.*

**Proof** By the chain rule, we have

$$\begin{aligned}\frac{\mathrm{d}L(X)}{\mathrm{d}X_R} &= \frac{\mathrm{d}L(X)}{\mathrm{d}Y_R}\frac{\mathrm{d}Y_R}{\mathrm{d}X_R} \\ &= \frac{\mathrm{d}L(X)}{\mathrm{d}Y_R}(I + \frac{\mathrm{d}\mathsf{H}(X_R)}{\mathrm{d}X_R}) \\ &= \frac{\mathrm{d}L(X)}{\mathrm{d}Y_R} + \frac{\mathrm{d}L(X)}{\mathrm{d}Y_R}\frac{\mathrm{d}\mathsf{H}(X_R)}{\mathrm{d}X_R} \end{aligned} \tag{32}$$

where the 1st step is from the chain rule, the 2nd step comes from basic calculus, the 3rd step is because of basic algebra.

By the assumption, we already have $\frac{\mathrm{d}L(X)}{\mathrm{d}Y_R}$, and $\frac{\mathrm{d}L(X)}{\mathrm{d}Y_R}\frac{\mathrm{d}\mathsf{H}(X_R)}{\mathrm{d}X_R}$ can be computed in almost linear time $n^{1+o(1)}$.

The addition operation between $\frac{\mathrm{d}L(X)}{\mathrm{d}Y_R}$ and $\frac{\mathrm{d}L(X)}{\mathrm{d}Y_R}\frac{\mathrm{d}\mathsf{H}(X_R)}{\mathrm{d}X_R}$ takes $n \cdot d$ time.

Therefore, the overall running time for $\frac{\mathrm{d}L(X)}{\mathrm{d}X_R}$ is $n^{1+o(1)}$.

Then, we consider the approximation error.

By Eq. (32) and basic linear algebra, the approximation error will not be magnified by more than $(n \cdot d \operatorname{poly}(n) + 1)$. Since $(n \cdot d \operatorname{poly}(n) + 1)(1/\operatorname{poly}(n)) = \operatorname{poly}(n)$, the approximation error on $\frac{\mathrm{d}L(X)}{\mathrm{d}X_R}$ can be bounded by $1/\operatorname{poly}(n)$.

■

### M.3. Analysis for the entire model with the residual connection

In the previous section, we have shown that, with the addition of the residual connection on a single component, the gradient computation time can still be done in almost linear time. We will apply this finding to the entire model.

We begin by single layer proof.

**Lemma 86 (Fast gradient computation for single-layer transformer with residual connection)** *If we have the below conditions,*

- *Let $L(X)$ be defined as Definition 5.*
- *Let $X \in \mathbb{R}^{n \times d}$ be defined as Definition 2.*
- *Suppose we have a single-layer transformer (see Definition 3).*
- *Let the residual connection be defined as Definition 83 and 84.*

*Then, we can show that,*

- ***Part 1: running time.** Our algorithm can approximate $\frac{\mathrm{d}L(X)}{\mathrm{d}X}$ in $n^{1+o(1)}$ time.*
- ***Part 2: error bound.** The approximation error of the single-layer transformer with the residual connection can be bounded by $1/\operatorname{poly}(n)$. Namely, our algorithm output $\widetilde{g}_{r_1}$ satisfies*

$$\|\widetilde{g}_{r_1} - \frac{\mathrm{d}L(X)}{\mathrm{d}X}\|_\infty \leq 1/\operatorname{poly}(n)$$

**Proof**

We use $T_i$ to represent $T_i(X)$ for simplicity. By the definition of $T_i$ (see also Definition 6), we have the following equations

$$T_0 = g_0(X)$$

Follow Definition 83 and 84, we have

$$Z_1 = T_0 + \mathsf{Attn}_1(T_0)$$

and

$$T_1 = Z_1 + g_1(Z_1)$$

Then we calculate the gradient by the following steps:

- **Step 1: Calculate** $\frac{\mathrm{d}L(X)}{\mathrm{d}T_1}$**.** By the definition of $L(X)$ (see also Definition 5), we have $\frac{\mathrm{d}L(X)}{\mathrm{d}T_1}$ can be computed in $n \cdot d$ time.
- **Step 2: Calculate** $\frac{\mathrm{d}L(X)}{\mathrm{d}Z_1}$**.** By Lemma 72, the assumption in Lemma 85 is satisfied. Therefore, we have $\frac{\mathrm{d}L(X)}{\mathrm{d}Z_1}$ can be computed in almost linear time $n^{1+o(1)}$.
- **Step 3: Calculate** $\frac{\mathrm{d}L(X)}{\mathrm{d}T_0}$**.** By Lemma 61, the assumption in Lemma 85 is satisfied. Hence, $\frac{\mathrm{d}L(X)}{\mathrm{d}T_0}$ can be computed in almost linear time. By Lemma 61, the approximation error is $1/\operatorname{poly}(n)$.
- **Step 4: Calculate** $\frac{\mathrm{d}L(X)}{\mathrm{d}X}$**.** By Lemma 72, $\frac{\mathrm{d}L(X)}{\mathrm{d}X}$ can be computed in $n^{1+o(1)}$. The approximation error is $(n \cdot d)(1/\operatorname{poly}(n)) = (1/\operatorname{poly}(n))$.

To sum up, we can show that the overall running time for $\frac{\mathrm{d}L(X)}{\mathrm{d}X}$ is $n^{1+o(1)}$ and the approximation error is $1/\operatorname{poly}(n)$.

Let $\widetilde{g}_{r_1}$ be the output of **Step 4**. Then we are done.

■

We now prove for multi-layer.

**Lemma 87 (Fast gradient computation for multi-layer transformer with residual connection)** *If we have the below conditions,*

- *Let $L(X)$ be defined as Definition 5.*
- *Let $X \in \mathbb{R}^{n \times d}$ be defined as Definition 2.*
- *Let the residual connection be defined as Definition 83 and 84.*
- *Suppose we have a $m$-layer transformer (see Definition 3).*

*Then, we can show that,*

- ***Part 1: running time.** Our algorithm can approximate $\frac{\mathrm{d}L(X)}{\mathrm{d}X}$ in $n^{1+o(1)}$ time.*

- ***Part 2: error bound.*** *The approximation error of the $m$-layer transformer with the residual connection can be bounded by $1/\operatorname{poly}(n)$. Namely, our algorithm output $\widetilde{g}_r$ satisfies*

$$\|\widetilde{g}_r - \frac{\mathrm{d}L(X)}{\mathrm{d}X}\|_\infty \leq 1/\operatorname{poly}(n)$$

**Proof** We use math induction in this proof.

**Step 1: Proof of a single-layer transformer.**

Firstly, by Lemma 86, we have the statement holds for a single-layer transformer.

**Step 2: Assumption for $k$-layer transformer.**

Secondly, we assume for any $k$, for $k$-layer transformer model, we have

- **Part 1: running time.** Our algorithm can approximate $\frac{\mathrm{d}L(X)}{\mathrm{d}X}$ in $O(n^{1+o(1)})$ time.
- **Part 2: error bound.** The approximation error of the $k$-layer transformer can be bounded by $1/\operatorname{poly}(n)$. Namely, our algorithm output $\widetilde{g}$ satisfies

$$\|\widetilde{g} - \frac{\mathrm{d}L(X)}{\mathrm{d}X}\|_\infty \leq 1/\operatorname{poly}(n)$$

**Step 3: Proof of $(k+1)$-layer transformer.**

Thirdly, we consider the $(k+1)$-layer transformer model.

Let $\mathsf{F}_k$ denote a $k$-layer transformer with the residual connection.

Then, the entire model can be written as

$$(\mathsf{F}_k \circ g_0)(X)$$

By the definition of $T_i$, we have

$$T_0 = g_0(X)$$

Then, by definition of $Z_i$ (see also Definition 83), we have

$$Z_1 = T_0 + \mathsf{Attn}_1(T_0)$$

By Definition 84, we have

$$T_1 = Z_1 + g_1(Z_1)$$

Without loss of generality, we assume that the additional transformer layer is added at the beginning of the model. Then, the $(k+1)$-layer transformer model has the following structure:

$$\mathsf{F}_{k+1}(X) = \mathsf{F}_k(T_1)$$

By the assumption for $k$-layer transformer, we have $\frac{\mathrm{d}L(X)}{\mathrm{d}T_1}$ can be computed in almost linear time $n^{1+o(1)}$ and the approximation error can be bounded by $1/\operatorname{poly}(n)$.

We apply similar proof of Lemma 86, then we can show that, we can compute $\frac{\mathrm{d}L(X)}{\mathrm{d}X}$ in almost linear time $n^{1+o(1)}$ and the approximation error can be bounded by $1/\operatorname{poly}(n)$.

■

## Appendix N. Multi-head Attention

Following the notation used in Section E.1, we use $h$ to denote the number of heads, and $d_h = d/h$ to denote the dimension of each head.

**Definition 88 (Multi-head attention)** *If we have the below conditions,*

- *Let $h$ denote the number of heads.*
- *Let $d$ denote the hidden dimension. Let $d_h = d/h$ denote the dimension of each attention head.*
- *Let $Q, K, V \in \mathbb{R}^{n \times d}$ be defined as Definition 2.*
- *Let $f(X)$ be defined as Definition 22.*
- *Let $s(X)$ be defined as Definition 24.*

*The multi-head attention can be formalized as follows:*

- ***Step 1.** Split the hidden dimension $d$ of $Q, K, V \in \mathbb{R}^{n \times d}$ into $h$ parts. Then, for each $l \in [h]$, we have $Q_l, K_l, V_l \in \mathbb{R}^{n \times d_h}$.*
- ***Step 2.** For each $l \in [h]$, calculate the attention matrix $f_l := \mathsf{Softmax}(Q_l K_l^\top / d_h) \in \mathbb{R}^{n \times n}$, and calculate the corresponding attention result $s_l := f_l V_l \in \mathbb{R}^{n \times d_h}$.*
- ***Step 3.** Concatenate $s_l \in \mathbb{R}^{n \times d_h}$ together, then we have the final multi-head attention output $s \in \mathbb{R}^{n \times d}$.*

Then, we dive into the analysis of the gradient computation process over the attention mechanism with multi-head attention.

**Lemma 89 (Analysis of the multi-head attention)** *If we have the below conditions,*

- *Let $\mathsf{Attn}(X)$ be defined as Definition 2.*
- *Let multi-head attention mechanism be defined as Definition 88.*
- *Let $Y_m, X_m \in \mathbb{R}^{n \times d}$ denote the output and input of the multi-head attention, respectively.*

*Then, we can show that,*

- *$\frac{\mathrm{d}L(X)}{\mathrm{d}X_m}$ can be calculated in almost linear time $n^{1+o(1)}$.*
- *If $\frac{\mathrm{d}L(X)}{\mathrm{d}Y_m}$ has $1/\operatorname{poly}(n)$ approximation error, then the approximation error on $\frac{\mathrm{d}L(X)}{\mathrm{d}X_m}$ is still $1/\operatorname{poly}(n)$.*

**Proof**

Following the notations used in Definition 88, for $l \in [h]$, we use $s_l \in \mathbb{R}^{n \times d_h}$ to denote the output by each attention head. And we use $s \in \mathbb{R}^{n \times d}$ to denote the concatenated version of the output of the multi-head attention.

By the chain rule and the definition of $L(X)$ (see also Definition 5), we have

$$\begin{aligned}\frac{\mathrm{d}L(X)}{\mathrm{d}X_m} &= \frac{\mathrm{d}L(X)}{\mathrm{d}Y_m} \cdot \frac{\mathrm{d}Y_m}{\mathrm{d}s}\frac{\mathrm{d}s}{\mathrm{d}X_m} \\ &= \frac{\mathrm{d}L(X)}{\mathrm{d}Y_m} \cdot \frac{\mathrm{d}Y_m}{\mathrm{d}s}\sum_{l=1}^{h}\frac{\mathrm{d}s_l}{\mathrm{d}X_m}\end{aligned}$$

where the 1st step is from the chain rule, the 2nd step comes from $s \in \mathbb{R}^{n\times d}$ is the concatenated version of $s_l \in \mathbb{R}^{n\times d_h}$.

We calculate the gradient in the following steps:

- **Step 1: Calculate $\frac{\mathrm{d}L(X)}{\mathrm{d}Y_m}$.** By the definition of $L(X)$ (Definition 5), we have that $\frac{\mathrm{d}L(X)}{\mathrm{d}Y_m}$ can be calculated in $n \cdot d$ time.
- **Step 2: Calculate $\frac{\mathrm{d}L(X)}{\mathrm{d}Y_m} \cdot \frac{\mathrm{d}Y_m}{\mathrm{d}s}$.** Since we already have $\frac{\mathrm{d}L(X)}{\mathrm{d}Y_m}$, by Lemma 72, we have $\frac{\mathrm{d}L(X)}{\mathrm{d}Y_m} \cdot \frac{\mathrm{d}Y_m}{\mathrm{d}s}$ can be computed in almost linear time $n^{1+o(1)}$.
- **Step 3: Calculate $\frac{\mathrm{d}L(X)}{\mathrm{d}Y_m} \cdot \frac{\mathrm{d}Y_m}{\mathrm{d}s}\sum_{l=1}^{h}\frac{\mathrm{d}s_l}{\mathrm{d}X_m}$.** For each $l \in [h]$, by Lemma 61, $\frac{\mathrm{d}L(X)}{\mathrm{d}Y_m} \cdot \frac{\mathrm{d}Y_m}{\mathrm{d}s} \cdot \frac{\mathrm{d}s_l}{\mathrm{d}X_m}$ can be computed in $n^{1+o(1)}$. Since the number of heads $h$ can be viewed as a constant here, it takes $n^{1+o(1)}$ time to compute the gradients on $h$ heads.

Therefore, the overall running time for $\frac{\mathrm{d}L(X)}{\mathrm{d}X_m}$ is $n^{1+o(1)}$.

Then, we consider the error bound.

By assumption, there is $1/\operatorname{poly}(n)$ approximation error on $\frac{\mathrm{d}L(X)}{\mathrm{d}Y_m}$. For each $l \in [h]$, the approximation error will not be magnified by more than $n^2 \cdot d \cdot d_h \cdot \operatorname{poly}(n)$ on $\frac{\mathrm{d}L(X)}{\mathrm{d}Y_m} \cdot \frac{\mathrm{d}Y_m}{\mathrm{d}s} \cdot \frac{\mathrm{d}s_l}{\mathrm{d}X_m}$.

Then, since there is total $h$ heads, the approximation error on $\frac{\mathrm{d}L(X)}{\mathrm{d}X_m}$ can be bound by

$$h \cdot n^2 \cdot d \cdot d_h \cdot \operatorname{poly}(n) \cdot (1/\operatorname{poly}(n)) = 1/\operatorname{poly}(n)$$

■

Similar to the proof of Lemma 73 and 74, we apply Lemma 89 to deal with the multi-head attention in each transformer layer. Then, we can show that $\frac{\mathrm{d}L(X)}{\mathrm{d}X}$ can be computed in almost linear time $n^{1+o(1)}$ and the approximation error can be bounded by $1/\operatorname{poly}(n)$.