Conv-Basis: A New Paradigm for Efficient Attention Inference and Gradient Computation in Transformers

Yingyu Liang[1,2] **Heshan Liu**[3] **Zhenmei Shi**[2]
Zhao Song[4,*] **Zhuoyan Xu**[2] **Jiale Zhao**[5] **Zhen Zhuang**[6]
[1]University of Hong Kong, [2]University of Wisconsin-Madison
[3]Carnegie Mellon University, [4]University of California, Berkeley
[5]Guangdong University of Technology, [6]University of Minnesota, Twin Cities
[*]magic.linuxkde@gmail.com

Abstract

The self-attention mechanism is key to the success of transformers in recent large language models (LLMs). However, the quadratic computational cost, $O(n^2)$, with respect to the input sequence length n poses a significant obstacle to further improvement and scalability in longer contexts. In this work, we leverage the convolution-like structure of attention matrices to develop an efficient approximation method for attention computation using convolution matrices. We propose a conv basis system, analogous to the rank basis, and show that any lower triangular matrix can be decomposed as a sum of structured convolution matrices in this basis. We then design a fast algorithm to approximate the attention matrix using a sum of k convolution matrices. This enables us to compute attention during *inference* via Fast Fourier Transforms (FFT) in $O(knd \log n)$ time, where d is the hidden dimension, achieving nearly linear time complexity, $n^{1+o(1)}$, in practical scenarios where $kd = n^{o(1)}$. Furthermore, both *training forward* and *backward gradient* computations can be performed in $n^{1+o(1)}$ time as well. We provide theoretical guarantees on runtime and approximation error and conduct preliminary experiments to evaluate the effectiveness of our approach. We hope this new paradigm for accelerating attention computation in transformer models facilitates their application to longer contexts.

1 Introduction

Numerous notable large language models (LLMs) in natural language processing (NLP) have emerged over the past two years, including Mistral (Jiang et al., 2023), Gemini (Team et al., 2023), Claude3 (Anthropic, 2024), GPT-4 (Achiam et al., 2023), and Llama3 (AI, 2024), among others. These models have profoundly impacted the world and are widely used in various domains, such as education (Kasneci et al., 2023), law (Sun, 2023), finance (Li et al., 2023a), bioinformatics (Thirunavukarasu et al., 2023), coding (Hou et al., 2024), and even creative writing (Achiam et al., 2023), including top AI conference reviews (Liang et al., 2024c).

The key to the success of generative LLMs is the decoder-only transformer architecture introduced by Vaswani et al. (2017). The transformer employs a self-attention mechanism, enabling the model to capture long-range dependencies in sequences. Self-attention computes a weighted sum of input tokens, where the weights are determined by the similarity between each token pair. This allows the model to focus on relevant information across different parts of the sequence when generating output. However, the computational complexity of self-attention grows quadratically, $O(n^2)$, with the input length n, limiting their scalability for long contexts. For example, GPT-4 (Achiam et al., 2023), Claude3 (Anthropic, 2024), and Gemma (Team et al., 2024) support input lengths of 128k, 200k, and 1000k tokens, respectively.

The complexity $O(n^2)$ arises from computing the similarity between each pair of tokens, resulting in an $n \times n$ matrix. More specifically, let d be the hidden dimension, and let $Q, K \in \mathbb{R}^{n \times d}$ be the query and key matrices of the input. Attention then requires computing the Softmax function on $QK^\top \in \mathbb{R}^{n \times n}$. Although $QK^\top$ has at most rank d, the Softmax operation in attention can produce $\mathsf{Softmax}(QK^\top) \in \mathbb{R}^{n \times n}$ with full rank.

To overcome the computational obstacle of $\mathsf{Softmax}(QK^\top)$, many studies have proposed more efficient attention computation methods that scale gracefully with sequence length while maintaining the model's performance. Alman and Song (2023) shows that if all entries of $QK^\top$ are bounded and $d = O(\log n)$, $\mathsf{Softmax}(QK^\top)$ will be "close" to a low-rank matrix. They then present an algorithm that approximates attention computation in almost linear time. Under the assumptions

Findings of the Association for Computational Linguistics: EMNLP 2025, pages 6857–6894
November 4-9, 2025

of uniform Softmax column norms and sparsity, Han et al. (2024) solves the attention computation in almost linear time by identifying large entries in the attention matrix and focusing only on them.

Another line of work (Olsson et al., 2022; Song and Zhong, 2023; Nichani et al., 2024; Reddy, 2024) finds that the attention pattern exhibits a convolutional-like (or "diagonalized") structure (see Figure 1 (b)). Mathematically, $A_{i,j} \approx A_{i',j'}$ when $i - j = i' - j'$, where $i - j$ represents the positional distance between two tokens. This is related to the bag-of-words or n-gram concept, i.e., sequences of n adjacent symbols or words in NLP. Furthermore, the convolutional-like structure connects to convolutional recurrent models (Bai et al., 2018), Hyena Hierarchy models (Poli et al., 2023; Massaroli et al., 2023), and structured state space models (SSMs) such as Mamba (Gu and Dao, 2023). More specifically, multiple convolution matrices can be used to approximate an attention matrix, following an intuition similar to low-rank approximation in terms of computational acceleration. Notably, the product of a convolution matrix and a vector can be computed using the Fast Fourier Transform (FFT) in $O(n \log n)$ time, whereas the naive approach requires $O(n^2)$ time (see details in Figure 1 (a)). Therefore, it is natural to ask:

Can we exploit the convolutional structure to accelerate the attention computation?

In this paper, we use multiple convolution matrices to efficiently approximate attention computation. Informally speaking, we present the following results, which apply to *any* $Q, K \in \mathbb{R}^{n \times d}$.

Theorem 1.1 (Main result, informal version of Theorem 3.4)**.** *Let $\epsilon > 0$, $k \in [n]$, and $Q, K \in \mathbb{R}^{n \times d}$. If $QK^\top$ is ϵ-close in the ℓ_∞ norm to a matrix with a k-conv basis (Definition 3.1), then the Exact Attention Computation (Definition 2.3) can be solved in $O(knd \log n)$ time via FFT, with an error of at most $O(\epsilon)$.*

When $kd = n^{o(1)}$, our method achieves an almost linear runtime of $n^{1+o(1)}$. Similar to low-rank approximation, we construct a conv basis system, analogous to the rank basis, and show that any lower triangular matrix $H \in \mathbb{R}^{n \times n}$ can always be decomposed into a k-conv basis for some $k \in [n]$, where $[n] = 1, 2, \ldots, n$ (Lemma 2.12 and Theorem 3.3). Then, Algorithm 2 efficiently decomposes $QK^\top$ into k convolution matrices when $QK^\top$ satisfies certain non-degeneracy properties (see Definition 3.1). Finally, using FFT, we achieve a time complexity of $O(knd \log n)$ to solve the task (Algorithm 1 and Theorem 3.4), whereas naive methods require $O(n^2 d)$.

Thus, our algorithm enables attention inference in $O(knd \log n)$ without any parameter updates, such as retraining or fine-tuning. Additionally, our theorems apply to accelerating attention training, requiring $O(knd \log n + nd^2)$ time for forward computation and $O(knd^2 \log n)$ time for backward gradient computation (Theorem 4.4). We also conduct preliminary experiments to evaluate its effectiveness (Section 5). Furthermore, our technique extends the low-rank approximation of attention matrices (Alman and Song, 2023) to more general settings (Theorem D.5). Specifically, Alman and Song (2023) focuses only on attention approximation without an attention mask, whereas our approach applies to various types of attention masks, including the widely used causal attention mask (Definition 2.2). This demonstrates the broad applicability of our analysis.

Our contributions. Our contributions are summarized as follows:

- We propose a conv basis system and show that any lower triangular matrix $H \in \mathbb{R}^{n \times n}$ can always be decomposed into a k-conv basis for some $k \in [n]$ (Lemma 2.12 and Theorem 3.3).
- We introduce an algorithm (Algorithm 2) that efficiently decomposes any lower triangular matrix into its k convolution basis. Using FFT, we solve the Exact Attention Computation task in $O(knd \log n)$ time (Algorithm 1 and Corollary 3.5). When $kd = n^{o(1)}$, our method runs in nearly linear time, $n^{1+o(1)}$. Our results surpass or are comparable to previous works (see comparison below).
- During attention inference, our algorithm runs in $O(knd \log n)$ time without requiring any parameter updates, such as retraining or fine-tuning (Theorem 3.4). Leveraging convolution properties and Fourier analysis, our approach provides stronger theoretical guarantees than existing methods.
- During attention training, our method requires $O(knd \log n + nd^2)$ time for forward computation and $O(knd^2 \log n)$ time for backward gradient computation (Theorem 4.4).
- Our broadly applicable technique extends low-rank approximations of attention matrices and

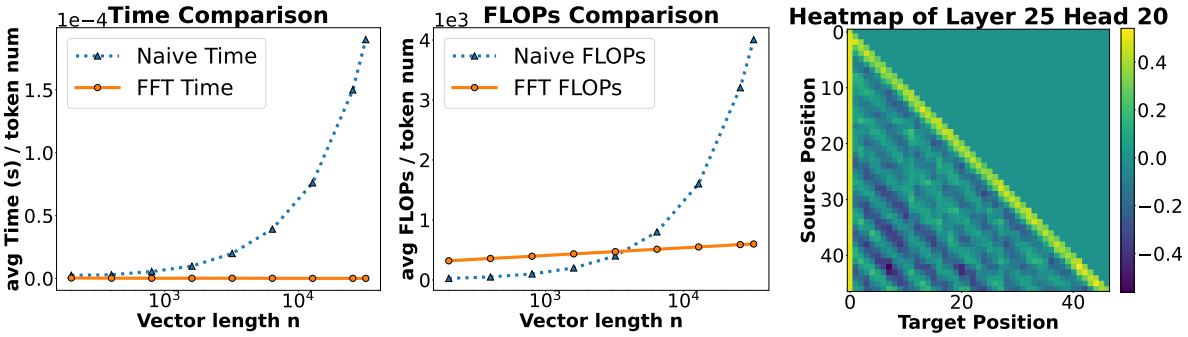

Figure 1: (a) In the left two figures, we compare the complexity of $\mathsf{conv}(a) \cdot w$ between the naive approach and FFT for a random vector $a, w \in \mathbb{R}^n$ and $\mathsf{conv}(a) \in \mathbb{R}^{n \times n}$ (Definition 2.5). The x-axis represents the input token number n, while the y-axis represents the average CPU time (first figure) and Float Operations (FLOPs) per token (second figure). The reported values are averaged over 100 runs using a NumPy implementation. It is evident that the naive approach takes $O(n^2)$ time, whereas the FFT approach runs in $O(n \log n)$. (b) In the right figure, we visualize an attention matrix $QK^\top \in \mathbb{R}^{n \times n}$ from Llama3 (AI, 2024), where the input is from SST-2 (Wang et al., 2018) with $n = 47$ tokens. The conv-like structure in the attention matrix is clearly observable.

generalizes existing results to more complex settings (Theorem D.5). Due to space limitations, we defer this result and its proof to the Appendix D.

Our motivation lies in addressing the limitations of prior methods that rely on restrictive assumptions. In contrast, we develop a theoretically grounded, assumption-light approach with provable guarantees. While our main focus is theoretical, we include experiments to demonstrate the method's practicality and ease of integration into real-world models like Llama 3.

Detailed comparison with previous works. Our results are beyond or comparable to the two brilliant previous works. (1) To ensure a small approximation error for the attention matrix, Alman and Song (2023) requires bounded entries and assumes $d = O(\log n)$, while Han et al. (2024) assumes uniform Softmax column norms and sparsity. However, our algorithm guarantees a small approximation error without any of these assumptions (Corollary 3.5). That is, our approach applies to any Q, K, including unbounded matrices, dense matrices, and any hidden dimension d. (2) Alman and Song (2023) assumes $d = O(\log n)$ to achieve a runtime of $n^{1+o(1)}$. In contrast, our algorithm achieves $n^{1+o(1)}$ runtime as long as $d = n^{o(1)}$ and $k = n^{o(1)}$, imposing far fewer restrictions on d. Moreover, our time complexity ranges from $n^{1+o(1)}$ to $n^{2-\Omega(1)}$ depending on d, whereas Alman and Song (2023) is limited to $d = O(\log n)$. (3) To ensure subquadratic runtime, Han et al. (2024) assumes $dm = n^{2-\Omega(1)}$, since their time complexity is $O(dn^{1+o(1)} + dm)$, where m represents the number of large entries in the attention matrix. Our approach has a time complexity of $O(knd \log n)$ and requires $kd = n^{1-\Omega(1)}$ for truly subquadratic runtime. In the case where $m = n^{1+o(1)}$, $d = n^{o(1)}$, and $k = n^{o(1)}$, both our algorithm and Han et al. (2024) achieve $n^{1+o(1)}$ runtime. However, if $m = n^{1+\Omega(1)}$, $d = n^{o(1)}$, and $k = n^{o(1)}$, Han et al. (2024) incurs a superlinear runtime of $n^{1+\Omega(1)}$, whereas our algorithm remains nearly linear at $n^{1+o(1)}$[1].

1.1 Related Work

Attention matrix conv-like structure. Very recent works study the conv-like attention matrix. Elhage et al. (2021); Olsson et al. (2022) find that in-context learning is driven by the formation of "induction heads"–attention heads that copy patterns from earlier in the input sequence. This is reflected in the attention matrix becoming more diagonal, with tokens attending primarily to preceding tokens that match the current token. In Song and Zhong (2023) Figure 6, they show a similar conv-like attention pattern for other important attention circuits. Figure 3 of Reddy (2024) shows that in a minimal classification task, the abrupt emergence of in-context learning coincides with the formation of an induction head, characterized by a diagonal attention pattern. Beltagy et al. (2020) proposed a

[1]For example, if the attention matrix is a lower triangular matrix with all entries equal to 1, we have $k = 1$ and $m = n(n+1)/2$.

combination of local and global attention, where local attention captures short-range dependencies, and global attention captures long-range dependencies. Wang et al. (2020) introduced a low-rank approximation of the self-attention mechanism that significantly reduces computational and memory costs while maintaining comparable performance to standard transformers. Nichani et al. (2024) proves that for a simplified task, gradient descent causes a transformer to encode the causal graph structure of the task in the attention matrix. This results in tokens attending primarily to their causal parents reflected in a sparse diagonal structure (Figure 2). In Li et al. (2024a), the conv-like attention matrix can also be observed when learning math tasks. Moreover, Cai et al. (2024) uses convolutional kernels to compress the KV-cache size for fast LLM generation.

We provide additional comparisons between our method and previous works in Section 6 for further reference.

2 Preliminaries

Notation. $\circ$ denotes element-wise multiplication. We denote $[n] = \{1, 2, \dots, n\}$ and $[0]$ as an empty set. $\mathbf{0}_n$ and $\mathbf{1}_n$ are n-dimensional vectors whose entries are all 0 and 1 respectively. $\exp(\cdot)$ is the element-wise exponential function. We denote $[x_a, x_{a+1}, \dots, x_b]^\top \in \mathbb{R}^{b-a+1}$ as $x_{a:b}$, where $1 \leq a \leq b \leq n$, similarly for matrix. $\mathrm{diag} : \mathbb{R}^n \to \mathbb{R}^{n \times n}$ is defined as $\mathrm{diag}(x)_{i,i} = x_i$ and $\mathrm{diag}(x)_{i,j} = 0$, for all $i \neq j$. For a matrix $A \in \mathbb{R}^{m \times n}$, its ℓ_1 norm is $\|A\|_1 = \sum_{i=1}^m \sum_{j=1}^n |A_{ij}|$, ℓ_∞ norm is $\|A\|_\infty = \max_{i,j} |A_{ij}|$, and Frobenius norm is $\|A\|_F := \sqrt{\sum_{i,j} A_{i,j}^2}$, where A_{ij} is an entry at the i-th row and j-th column. $\mathcal{T}_{\mathrm{mat}}(n, d, k)$ is the time of an $n \times d$ matrix times a $d \times k$ matrix. Given two matrices $A_1 \in \mathbb{R}^{n_1 \times d_1}$, $A_2 \in \mathbb{R}^{n_2 \times d_2}$, for all $i_1 \in [n_1], i_2 \in [n_2], j_1 \in [d_1], j_2 \in [d_2]$, we define $A := A_1 \otimes A_2 \in \mathbb{R}^{n_1 n_2 \times d_1 d_2}$ as $A_{i_1 + (i_2 - 1) n_1, j_1 + (j_2 - 1) d_1} = (A_1)_{i_1, j_1} \cdot (A_2)_{i_2, j_2}$.

2.1 Basic Definitions and Facts

Now, we present basic definitions. We start by introducing the input and weight matrix.

Definition 2.1 (Input and weight matrix)**.** *We define the input sequence as $X \in \mathbb{R}^{n \times d}$ and the key, query, and value weight matrix as $W_K, W_Q, W_V \in \mathbb{R}^{d \times d}$. Then, we define the key, query, and value matrix as $K := XW_K \in \mathbb{R}^{n \times d}$, $Q := XW_Q \in \mathbb{R}^{n \times d}$, $V := XW_V \in \mathbb{R}^{n \times d}$.*

Note that $QK^\top = XW_Q W_K^\top X^\top$. In LLMs, there is a causal mask M to guarantee the later tokens cannot see the prior tokens during generation.

Definition 2.2 (Causal attention mask)**.** *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. We define M_j to be the j-th column of M.*

Now, we introduce the mathematical definition of the exact attention computation with a mask.

Definition 2.3 (Exact attention computation)**.** *Let $Q, K, V \in \mathbb{R}^{n \times d}$ be the query, key, and value matrices respectively defined in Definition 2.1. Let $M \in \{0, 1\}^{n \times n}$ be the attention mask defined in Definition 2.2. The goal of the Exact Attention Computation is to find the matrix $\mathrm{Att}(M, Q, K, V) \in \mathbb{R}^{n \times d}$, which is defined as $\mathrm{Att}(M, Q, K, V) := D^{-1}AV$ where $A \in \mathbb{R}^{n \times n}$ is a lower triangular matrix and $D \in \mathbb{R}^{n \times n}$ is a diagonal matrix, i.e., $A := M \circ \exp(QK^\top)$ and $D := \mathrm{diag}(A\mathbf{1}_n)$.*

Remark 2.4. *In Definition 2.3, we divide the* Softmax *operation into an element-wise* exp *operation and a diagonal normalization matrix D to obtain a clear formulation.*

Efficiently computing the attention needs to exploit structured matrices that enable fast multiplication algorithms. Here, we define the convolution matrix, which is a structured matrix where each row vector is rotated one element to the right relative to the preceding row vector.

Definition 2.5 (Convolution matrix)**.** *Let $a \in \mathbb{R}^n$. We define $\mathsf{conv} : \mathbb{R}^n \to \mathbb{R}^{n \times n}$ as,*

$$\mathsf{conv}(a) := \begin{bmatrix} a_1 & 0 & 0 & \cdots & 0 \\ a_2 & a_1 & 0 & \cdots & 0 \\ a_3 & a_2 & a_1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_n & a_{n-1} & a_{n-2} & \cdots & a_1 \end{bmatrix}.$$

By the following fact, we know that the rank of a convolution matrix can be an arbitrary number. Thus, our conv-basis is totally different from the rank basis. See proof in Appendix B.1.

Claim 2.6. *We have $\mathsf{conv}(e_j) \in \mathbb{R}^{n \times n}$ is a j-rank matrix, where the j-th entry of $e_j \in \mathbb{R}^n$ is 1 and all other entries are 0.*

Efficient computation of the convolution operation is crucial for many applications. The convolution theorem states that the circular convolution of two vectors can be computed efficiently using the Fast Fourier Transform (FFT). This leads to the following claim (see proof in Appendix B.1):

Claim 2.7. *Let* conv *be defined in Definition 2.5. For any $a, x \in \mathbb{R}^n$, $\mathsf{conv}(a)x$ can be computed in $O(n \log n)$ via FFT.*

One property of convolution matrices is that they are additive with respect to the input vectors. In other words, the convolution of the sum of two vectors is equal to the sum of the convolutions of the individual vectors. This is stated formally in the following claim (see proof in Appendix B.1):

Claim 2.8. conv *is additive, i.e., for any $a, b, x \in \mathbb{R}^n$ we have $\mathsf{conv}(a)x + \mathsf{conv}(b)x = \mathsf{conv}(a+b)x$.*

Many other interesting facts and properties about the convolution matrix are used in our main theorem proof. Due to space limitations, we leave them in Appendix B.1 for reader interests.

2.2 Sub-convolution Matrix: Definitions and Properties

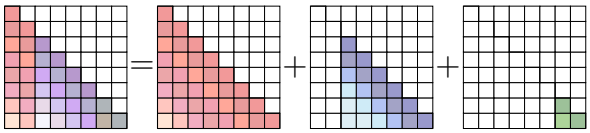

Figure 2: We present an example of the matrix defined in Definition 2.11 when $k = 3$. The matrix with 3-conv basis is on the left-hand side of the equation in this figure. The red entries in this matrix come from the first matrix on the right-hand side. The purple entries in this matrix are the sum of the red entries from the first matrix on the right-hand side and the blue entries from the second matrix on the right-hand side. The dark green entries are equal to the sum of red, green, and blue entries from the matrices on the right-hand side.

If we would like to use conv as a basis system, we need to introduce some new concepts. Recall that, in general, the sum of two rank-1 matrices is a rank-2 two matrix. Due to conv being additive, the sum of two convolution matrices is another convolution matrix, which does not hold the above property. Thus, we need to introduce sub-convolution matrices to be the basis.

Definition 2.9 (Sub-convolution matrix)**.** *Let $m \in [n]$. For any $a \in \mathbb{R}^n$. We define the sub-convolution matrix $\mathsf{conv}(a, m)$ as*

$$\mathsf{conv}(a, m) = \begin{bmatrix} \mathbf{0}_{(n-m)\times(n-m)} & \mathbf{0}_{(n-m)\times m} \\ \mathbf{0}_{m\times(n-m)} & \mathsf{conv}(a_{1:m}) \end{bmatrix}.$$

Given two vectors $a, x \in \mathbb{R}^n$, let $a *_m x \in \mathbb{R}^n$ denote the sub-convolution operator between a and x, i.e., $\mathsf{conv}(a, m)x = a *_m x$.

Similarly, sub-convolution can be computed in $O(n \log n)$ time via FFT (see proof in Appendix B.1).

Claim 2.10. *Let $m \in [n]$. For any $a, x \in \mathbb{R}^n$, $\mathsf{conv}(a, m)x$, (defined in Definition 2.9) can be computed in $O(n \log n)$ via FFT.*

Here, we present the definition of the matrix with k-conv basis which is non-reducible.

Definition 2.11 (Matrix with k-conv basis)**.** *Let $k \in [n]$. We say a lower triangular matrix $H \neq \mathbf{0}_{n\times n} \in \mathbb{R}^{n\times n}$ has k-conv basis if*

- *There exists $b_1, \ldots, b_k \in \mathbb{R}^n$ and k integers $m_1, m_2, \ldots, m_k$ satisfying $n \geq m_1 > m_2 > \cdots > m_k \geq 1$ such that $H = \sum_{i\in[k]} \mathsf{conv}(b_i, m_i)$, (Definition 2.9).*
- *For all $b_1, \ldots, b_{k-1} \in \mathbb{R}^n$ and integers $m_1, \ldots, m_{k-1}$ with $n \geq m_1 > m_2 > \cdots > m_{k-1} \geq 1$, $H \neq \sum_{i\in[k-1]} \mathsf{conv}(b_i, m_i)$.*

The following lemma establishes that any non-zero lower triangular matrix can be represented as a matrix with a k-conv basis for some unique k between 1 and n. The proof is in Appendix F.1.

Lemma 2.12. *For any lower triangular matrix $H \neq \mathbf{0}_{n\times n} \in \mathbb{R}^{n\times n}$, there exists a unique $k \in [n]$ such that H is a matrix with k-conv basis.*

3 conv Approximation during Inference

In Section 3.1, we introduce the basic definitions to support our algorithmic analysis in this section. In Section 3.2, we present the binary search and recover k-conv algorithms and present their theoretical guarantees. In Section 3.3, we provide the formal version of our main result.

3.1 Key Concepts

Any non-zero lower triangular matrix can be represented as a matrix with a k-conv basis for some unique k between 1 and n (Lemma 2.12). However, exactly getting k is hard and the definition is too strict for the algorithm design. Thus, for more flexibility, we introduce a more general definition of non-degenerate k-conv basis as below, which is a proxy notion to relax the conditions required.

Definition 3.1 (Non-degenerate k-conv basis)**.** *Let $T \in [n]$, $\delta \geq 0$, and $k \in [n + 1 - T]$. Let $b_1, \ldots, b_k \in \mathbb{R}^n$ and k integers $m_1, m_2, \ldots, m_k$ satisfying $n \geq m_1 > m_2 > \cdots > m_k \geq T$. Let $H = \sum_{i\in[k]} \mathsf{conv}(b_i, m_i)$. If for each basis $i \in [k]$,*

*for all $j \in [i]$, we have $\| \sum_{l=j}^{i} (b_l)_{1:T} \|_1 \geq \delta$, then we define $H \in \mathbb{R}^{n \times n}$ to be a matrix with (T, δ)-non-degenerate k-*conv *basis.*

Here (T, δ)-non-degenerate k-conv basis means that each conv basis cannot be "covered" by the other basis easily.

Definition 3.2. *We define G as a ϵ-close (T, δ)-non-degenerate k-*conv *basis matrix when $G = H + R$, where H is a (T, δ)-non-degenerate k-*conv *basis matrix defined in Definition 3.1 and the noise matrix $R \in \mathbb{R}^{n \times n}$ satisfies $\|R\|_\infty \leq \epsilon \leq \frac{\delta}{5T}$.*

The following theorem establishes that any non-zero lower triangular matrix can be represented as an ϵ-close (T, δ)-non-degenerate k-conv basis matrix (see proof in Section B.2). There may be many different choices of (k, T, δ, ϵ), which provide flexibility for our Algorithm 1.

Theorem 3.3. *For any lower triangular matrix $G \neq \mathbf{0}_{n \times n} \in \mathbb{R}^{n \times n}$, there exists $k, T \in [n]$ and $\delta, \epsilon \geq 0$ such that G is a ϵ-close (T, δ)-non-degenerate k-*conv *basis matrix.*

3.2 Algorithms and Their Properties

Now, we present our main Algorithm 1. We present Algorithm 2 and Algorithm 3 as well.

Algorithm 1 Main k-conv forward

procedure convFORWARD($Q, K, V \in \mathbb{R}^{n \times d}, k, T \in [n], \delta, \epsilon \in \mathbb{R}_{\geq 0}$) ▷ Theorem 3.4
  $\widetilde{b}_1, \ldots, \widetilde{b}_k, m_1, \ldots, m_k \leftarrow$ RECOVER($Q, K, k, T, \delta, \epsilon$) ▷ Algorithm 2, recover k-conv
  $\widetilde{D} \leftarrow \mathrm{diag}(\sum_{r \in [k]} \mathsf{conv}(\widetilde{b}_r, m_r) \mathbf{1}_n)$ by FFT in Claim 2.10
  $\widetilde{Y} \leftarrow \widetilde{D}^{-1} \sum_{r \in [k]} \mathsf{conv}(\widetilde{b}_r, m_r) V$ by FFT in Claim 2.10
  return $\widetilde{Y}$
end procedure

In Algorithm 1, we first using Algorithm 2 to get k conv basis. Then, we can get the approximated normalization matrix $\widetilde{D}$ and the final output $\widetilde{Y}$ by FFT in Claim 2.7.

In Algorithm 2, we iteratively use binary search (Algorithm 3) to identify the conv basis positions and compute their values. Note that, in the final step, we replace b'_i with $\widetilde{b}_i$ by incorporating the exp function used in Softmax. We will provide proofs of correctness and time complexity below.

In Algorithm 3, we employ binary search to efficiently locate the convolution basis positions

Algorithm 2 Recover k-conv

procedure RECOVER($Q, K \in \mathbb{R}^{n \times d}, k, T \in [n], \delta, \epsilon \in \mathbb{R}_{\geq 0}$)
  $v \leftarrow \mathbf{0}_T, u \leftarrow \mathbf{0}_n, s \leftarrow 0, t \leftarrow n - T + 1$ ▷ Initialize the state for binary search
  for $i = 1 \to k$ **do**
    $s \leftarrow s + 1$
    $s \leftarrow$ SEARCH($Q, K, k, T, \delta, \epsilon, v, s, t$) ▷ Algorithm 3 in Appendix B.2, binary search the next conv basis position
    $m_i \leftarrow n - s + 1$
    $\widetilde{H}_s \leftarrow M_s \circ (Q(K^\top)_s)$
    $(b'_i)_{1:m_i} \leftarrow \widetilde{H}_{s,s:s+m_i-1} - u_{1:m_i}$, $(b'_i)_{m_i+1:n} \leftarrow \mathbf{0}_{n-m_i}$ ▷ Get the conv basis value
    $v \leftarrow v + (b'_i)_{1:T}$
    $u \leftarrow u + b'_i$
  end for
  Get $\widetilde{b}_1, \ldots, \widetilde{b}_k$ by Lemma B.16 from $b'_1, \ldots, b'_k$ and $m_1, \ldots, m_k$
  return $\widetilde{b}_1, \ldots, \widetilde{b}_k, m_1, \ldots, m_k$
end procedure

by leveraging the non-degenerate property (Definitions 3.1 and 3.2) of the attention matrix. This property allows us to identify the k-conv basis in our main Algorithm 1, enabling better control over runtime while bounding the error. The choice of k balances the trade-off between accuracy and efficiency. Algorithm 3 identifies positions in the attention matrix where the ℓ_1 norm of the remaining attention values exceeds the threshold $\delta - 2T\epsilon$. The non-degenerate property ensures that the binary search algorithm can find the next convolution basis position in $O(\log n)$ steps.

3.3 Main Theoretical Result

In this section, we present our main result.

Theorem 3.4 (Main conv results for inference)**.** *Let $Q, K, V \in \mathbb{R}^{n \times d}$. Let $A = M \circ \exp(QK^\top) \in \mathbb{R}^{n \times n}$, $D = \mathrm{diag}(A \mathbf{1}_n) \in \mathbb{R}^{n \times n}$ be defined in Definition 2.3. We let $Y := D^{-1} A V \in \mathbb{R}^{n \times d}$. Let $M \circ (QK^\top)$ be an ϵ-close (T, δ)-non-degenerate k-*conv *basis matrix (Definition 3.2), where $\delta, \epsilon \geq 0$ and $k, T \in [n]$. By Algorithm 1, we can get $\widetilde{Y}$ with*

$$\|Y - \widetilde{Y}\|_\infty \leq 2(\exp(2\epsilon) - 1) \|V\|_\infty,$$

whose time complexity is $O(knd \log(n))$.

Proof sketch. See complete proof in Appendix B.4. The proof idea is that using binary search to re-

Algorithm 3 Binary search

procedure SEARCH($Q, K \in \mathbb{R}^{n \times d}, k, T \in [n], \delta, \epsilon \in \mathbb{R}_{\geq 0}, v \in \mathbb{R}^T, s, t \in [n]$)
  if $s \geq t$ **then**
    return s
  end if
  $j \leftarrow \lfloor (s+t)/2 \rfloor$
  $\widetilde{H}_j \leftarrow M_j \circ (Q(K^\top)_j)$ ▷ $j \in [n]$, M is attention mask defined in Definition 2.2
  $\alpha \leftarrow \|(\widetilde{H}_j)_{j:j+T-1} - v\|_1$
  if $\alpha \geq \delta - 2T\epsilon$ **then**
    return SEARCH($Q, K, k, T, \delta, \epsilon, v, s, j$)
  else
    return SEARCH($Q, K, k, T, \delta, \epsilon, v, j+1, t$)
  end if
end procedure

cover all non-degenerate conv basis (Lemma B.19), which takes $O(knd \log(n))$ time and has upto $2(\exp(2\epsilon) - 1)\|V\|_\infty$ error (Lemma B.20). Then, via FFT (Claim 2.10), we finish the proof. □

Note that our algorithm can handle any $Q, K \in \mathbb{R}^{d \times d}$. Furthermore, we can exactly recover Y if we do not care about the time complexity. We formally describe the above intuition as follows.

Corollary 3.5 (Exact conv inference)**.** *Let $Q, K, V \in \mathbb{R}^{n \times d}$. Recall $A = M \circ \exp(QK^\top) \in \mathbb{R}^{n \times n}$, $D = \mathrm{diag}(A\mathbf{1}_n) \in \mathbb{R}^{n \times n}$ defined in Definition 2.3. We denote $Y := D^{-1}AV \in \mathbb{R}^{n \times d}$. For any $\epsilon \geq 0$ and any Q, K, V, there exists a hyper-parameter $k, T \in [n]$ and $\delta \geq 0$ such that Algorithm 1 can output $\widetilde{Y}$ satisfying*

$$\|Y - \widetilde{Y}\|_\infty \leq 2(\exp(2\epsilon) - 1)\|V\|_\infty.$$

Furthermore, we can exactly get Y, i.e., $\epsilon = 0$, through Algorithm 1 with a time complexity $O(n^2 d \log(n))$ in the worst case.

We defer the proof to Appendix B.4. By Theorem 3.4, for $\epsilon = O(1)$, we get the attention inference time complexity is $O(knd \log(n))$ with error up to $O(\epsilon)$. It may enable further improvement and scalability of LLMs in the longer context. Moreover, in Section 6, we provide a detailed discussion about two case studies, LongLora (Chen et al., 2023b) and RoPE (Su et al., 2024), where our algorithm can apply to these two long-context LLMs as well. We also provide further discussion on limitations and extensions there.

4 conv Approximation for Training

We accelerate both forward and back propagation of attention. We first define attention optimization, which is also used in Alman and Song (2024a).

Definition 4.1 (Attention optimization)**.** *Given $A_1, A_2, A_3, E \in \mathbb{R}^{n \times d}$ and $Y \in \mathbb{R}^{d \times d}$, let $M \in \mathbb{R}^{n \times n}$ be a casual attention mask (Definition 2.2). Let $D(X) := \mathrm{diag}(M \circ \exp(A_1 X A_2^\top)\mathbf{1}_n) \in \mathbb{R}^{n \times n}$. The attention optimization is to find $\min_{X \in \mathbb{R}^{d \times d}} L(X)$, where $L(X)$ is defined as*

$$0.5\|D(X)^{-1}M \circ \exp(A_1 X A_2^\top)A_3 Y - E\|_F^2.$$

Remark 4.2. *Our Attention Optimization task in Definition 4.1 covers both the cross-attention and self-attention setting. Let weight matrices $W_K, W_Q, W_V \in \mathbb{R}^{d \times d}$ be defined in Definition 2.1. For the self-attention setting, we can see $A_1, A_2, A_3 \in \mathbb{R}^{n \times d}$ as $X \in \mathbb{R}^{n \times d}$ in Definition 2.1, see $X \in \mathbb{R}^{d \times d}$ in Definition 4.1 as $W_Q W_K^\top \in \mathbb{R}^{d \times d}$ and see $Y \in \mathbb{R}^{d \times d}$ as $W_V \in \mathbb{R}^{d \times d}$. To overcome the quadratic complexity obstacle, we only need to handle the gradient computation of $W_Q W_K^\top$.*

As $QK^\top \in \mathbb{R}^{n \times n}$ during inference, in gradient calculation, we have an $n \times n$ matrix $u(x)$.

Definition 4.3. *Let $M \in \mathbb{R}^{n \times n}$ be a casual attention mask defined in Definition 2.2. Let $A_1, A_2 \in \mathbb{R}^{n \times d}$. Suppose that $\mathsf{A} = A_1 \otimes A_2 \in \mathbb{R}^{n^2 \times d^2}$. For all $j_0 \in [n]$, let $\mathsf{A}_{j_0} \in \mathbb{R}^{n \times d^2}$ be the j_0-th block of A and $u(x)_{j_0} := M_{j_0,*} \circ \exp(\mathsf{A}_{j_0} x)$, where $x \in \mathbb{R}^{d^2}$ is the vectorization of $X \in \mathbb{R}^{d \times d}$. Define $u(x) \in \mathbb{R}^{n \times n}$ as the matrix where the j_0-th row corresponds to $(u(x)_{j_0})^\top$.*

We show our main result of attention training.

Theorem 4.4 (Main conv result for training forward and backward gradient)**.** *If $u(x)$ is a $1/\operatorname{poly}(n)$-close (T, δ)-non-degenerate k-conv basis matrix as defined in Definition 3.2, where $\delta \geq 0$ and $k, T \in [n]$. Then there are algorithms that run to compute **training forward** in time $O(knd \log n + \mathcal{T}_{\mathrm{mat}}(n, d, d))$ and **backward gradient** in time $O(d^2 kn \log n)$ of attention loss (Definition 4.1) approximately up to $1/\operatorname{poly}(n)$ error under ℓ_∞ norm.*

Proof sketch. See complete proof in Appendix C.4. During backward computation, we can convey the properties of low-rank and convolution at the same time (Lemma C.13 and Lemma C.15). Then, by tensor trick, we can compute the attention gradient

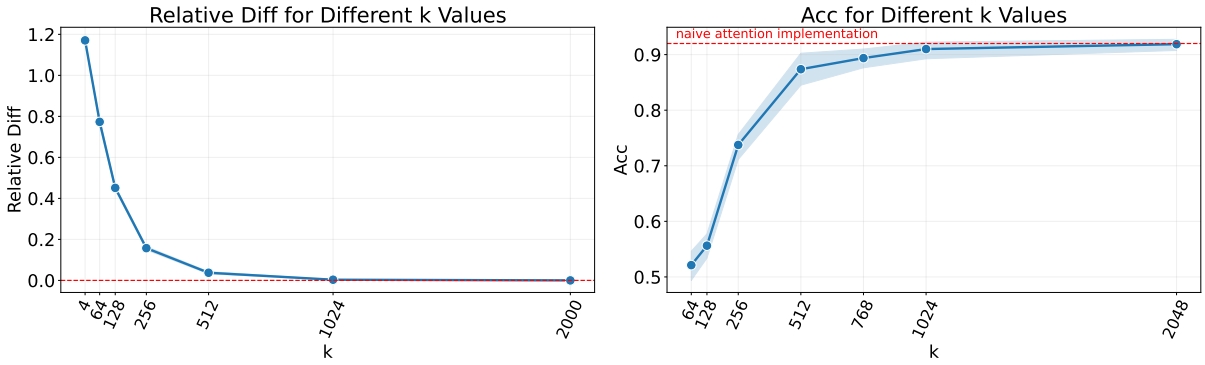

Figure 3: The comparison between the Llama3 8B Instruct with or without using our Algorithm 1 on the IMDB dataset. The input sequence length $n = 2048$. The x-axis is the number of conv basis. The y axis is relative difference $\frac{\|Y-\widetilde{Y}\|_F^2}{\|Y\|_F^2}$ for the left figure and classification accuracy for the right figure. Note that $k = 2048$ represents the baseline of the original model, as this is the input sequence length.

based on attention inference (Lemma C.9). We finish the proof by Theorem 3.4. □

Remark 4.5. *Alman and Song (2024a) only convey the low-rank property, but we convey both low-rank and convolution properties, which is general.*

The parameter k serves as a critical design parameter in our method. A larger k improves approximation accuracy but increases computation, while a smaller k reduces cost but may affect performance. Moreover, Theorem 4.4 shows k is insensitive to context length; as long as k grows sublinearly with input length n, the algorithm runs in sub-quadratic time. It also proves our method can accelerate Transformer training, demonstrating its practical advantage.

5 Experiments

Now, we show our experimental results for convolution attention computing in language models, offering empirical backing to our theoretical claims.

Setup. We use the latest Llama3 8B Instruct model[2] (AI, 2024) as our foundation, modifying its attention mechanism with our convolution-based approach using varying numbers of convolution bases k. We evaluate our method on the IMDB dataset (Maas et al., 2011), which consists of labeled movie reviews. Our assessments rely on two key metrics: (1) the relative difference between our final layer output $\widetilde{Y}$ and the original model output Y, i.e., $\|Y - \widetilde{Y}\|_F^2 / \|Y\|_F^2$; and (2) classification accuracy. This dual approach enables us to assess both internal representations and the overall predictive performance of our convolution-based attention compared to the standard mechanism.

Implementation details. To ensure a fair comparison and prevent memory issues, we set the model's context length to 2048 tokens and incrementally increased the number of conv bases k. When $k = 2048$, our convolution attention produces an identical output to the original attention mechanism. We use an instruction-based approach to evaluate generation accuracy, formatting our input as *Review: <REVIEW> Question: Is this review positive or negative? Answer:*. This technique allows us to systematically assess the performance of our convolution-based attention across various complexity levels while maintaining comparability with the original model. We randomly sample 5 sample groups, with 200 samples per group, and report the results averaged across each group. All evaluations are conducted by directly applying our approximation to the pretrained Llama3 8B Instruct model without any retraining or fine-tuning, isolating the effect of the attention modification.

Results. Figure 3 (left) shows that the relative MSE drops rapidly as k increases, even with small values like $k = 256$ or 512, indicating convergence toward the original attention. The right plot shows accuracy also improves with k and matches the original at $k = 512$, demonstrating that our method retains strong performance with lower computational costs. These results imply that our approach effectively approximates the original attention mechanism, offering a promising trade-off between accuracy and efficiency, particularly in resource-constrained scenarios.

[2] https://huggingface.co/meta-llama/Meta-Llama-3-8B-Instruct

6 Further Discussion

Comparison with efficient attention methods. Recent works such as Linformer (Wang et al., 2020) and Longformer (Beltagy et al., 2020) reduce attention complexity by imposing structural assumptions like low-rank projections or sparsity patterns on the attention matrix. In contrast, our approach leverages a convolutional basis decomposition, which does not rely on any low-rank hypothesis. The conv-basis framework is provably applicable to any lower triangular matrix, including masked attention scenarios. Importantly, our theoretical results (Theorem 3.4 and Corollary 3.5) guarantee that the approximation error is tightly controlled and can be made arbitrarily small by increasing k, with provable bounds on $\|Y - \widetilde{Y}\|_\infty$.

LongLora. Our conv and low-rank approximation can be applied to LongLora (Chen et al., 2023b), whose mask is shown in the left of Figure 4. They use this kind of sparse mask to extend the context sizes of pre-trained large language models, with limited computation cost, e.g., extending Llama2 70B from 4k context to 32k on a single $8\times$ A100 machine. As the "diagonalized" mask structure, we can directly apply our Algorithm 1 by replacing the causal attention mask (Definition 2.2) with their sparse mask for the conv approximation with time complexity $O(knd\log(n))$. Similarly, for the low-rank approximation, we directly use the second statement in Theorem D.5 by considering row change by amortized constant mask defined in Definition D.1 with time complexity $O(knd)$, where $B_j = O(1)$ for any $j \in [n]$.

RoPE. The Rotary Position Embedding (RoPE) (Su et al., 2024) designs a rotation matrix $R^{(m)} \in \mathbb{R}^{d\times d}$, for all $m \in [n]$, which can effectively encode the positional information into embedding $Q, K \in \mathbb{R}^{n\times d}$. In detail, let $q_i, k_j \in \mathbb{R}^d$, where $q_i^\top$ and $k_j^\top$ be the i-th and j-th row of Q, K respectively, for any $i, j \in [n]$. By the property of rotation matrix, we have

$$(R^{(i)}q_i)^\top (R^{(j)}k_j) = q_i^\top R^{(j-i)}k_j$$

We define $Q', K' \in \mathbb{R}^{n\times d}$, and let $q_i', k_j' \in \mathbb{R}^d$, where $q_i'^\top$ and $k_j'^\top$ be the i-th and j-th row of Q', K' respectively, for any $i, j \in [n]$. Let $q_i' = R^{(i)}q_i$ and $k_j' = R^{(j)}k_j$. By Equation (34) in (Su et al., 2024), we know that we can get Q', K' in $O(nd)$ time. Thus, we can apply Q', K' in our Theorem 3.4 and Theorem D.5 to get the same approximation error guarantee and the same time complexity.

Extend to full self-attention. We can easily extend our method to full self-attention. Our proposed approach can be extended to accelerate full self-attention as well, not just the causal attention mechanism. Note that the full self-attention matrix can be split into a lower triangular matrix L and an upper triangular matrix U. Then, our conv-basis approximation method can be applied separately to L and the transpose of U. This allows the algorithm to handle both the lower and upper triangular components of the full attention matrix. The diagonal normalization step D^{-1} would need to be adjusted to account for the full matrix rather than just the lower triangular portion. Finally, we combine the approximations of L and $U^\top$ to reconstruct the full self-attention output.

Memory consumption. Our method does not increase the memory consumption because each convolution matrix can be stored as a n-dimention vector (see Definition 2.5). Therefore, our method requires $O(kn)$ memory for k convolution matrices, $O(nd)$ memory for the value matrix $V \in \mathbb{R}^{n\times d}$, and $O(n)$ memory for the diagonal matrix $D \in \mathbb{R}^{n\times n}$. In total, our memory consumption is $O(kn + nd)$. For the standard attention computation of $D^{-1}AV$, it requires $O(n^2)$ memory for the attention matrix A, $O(nd)$ memory for the value matrix $V \in \mathbb{R}^{n\times d}$, and $O(n)$ memory for the diagonal matrix $D \in \mathbb{R}^{n\times n}$. In total, the memory consumption is $O(n^2 + nd)$.

7 Conclusion

We introduce a novel method for efficient attention computation in transformers by leveraging a structured convolution matrix decomposition. Unlike previous approaches that often rely on restrictive assumptions or focus exclusively on approximating specific types of attention, our algorithm provides a general, assumption-light framework with provable guarantees. It achieves nearly linear time complexity for both attention inference and gradient computation, offering stronger theoretical assurances than existing methods. This work establishes a new paradigm for accelerating attention operations, enabling transformers to handle longer contexts more efficiently and paving the way for further advancements in Large Language Models.

Limitation

We focus primarily on theoretical analysis, with relatively limited empirical validation. Although our paper includes preliminary experiments on the IMDB dataset, more comprehensive real-world benchmarking across diverse tasks and model architectures would further validate the practical benefits of the proposed conv-basis approach. However, we believe that our extensive theoretical development outweighs this experimental limitation. We provide a detailed comparison of our work with recent papers, such as Alman and Song (2023); Han et al. (2024), in Section 1. Our motivation stems from the fact that their assumptions are overly restrictive, making their algorithms difficult to implement successfully. In contrast, our work develops an implementable, assumption-light algorithm that can be applied in real-world scenarios while maintaining a comparable running time to Alman and Song (2023); Han et al. (2024). The experiment in our paper serves as an example to demonstrate the implementation of our theoretical contribution and to highlight its independence from the restrictive assumptions of Alman and Song (2023), which is a purely theoretical work with no experiments.

References

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

Kwangjun Ahn, Xiang Cheng, Minhak Song, Chulhee Yun, Ali Jadbabaie, and Suvrit Sra. 2024. Linear attention is (maybe) all you need (to understand transformer optimization). In *The Twelfth International Conference on Learning Representations*.

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

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

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

Josh Alman and Zhao Song. 2024b. How to capture higher-order correlations? generalizing matrix softmax attention to kronecker computation. In *The Twelfth International Conference on Learning Representations*.

Josh Alman and Zhao Song. 2025a. Fast rope attention: Combining the polynomial method and fast fourier transform. *arXiv preprint arXiv:2505.11892*.

Josh Alman and Zhao Song. 2025b. Only large weights (and not skip connections) can prevent the perils of rank collapse. *arXiv preprint arXiv:2505.16284*.

Chenxin An, Fei Huang, Jun Zhang, Shansan Gong, Xipeng Qiu, Chang Zhou, and Lingpeng Kong. 2024. Training-free long-context scaling of large language models. *arXiv preprint arXiv:2402.17463*.

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

Shaojie Bai, J Zico Kolter, and Vladlen Koltun. 2018. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*.

Björn Bebensee and Haejun Lee. 2023. Span-selective linear attention transformers for effective and robust schema-guided dialogue state tracking. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*.

Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

Amanda Bertsch, Uri Alon, Graham Neubig, and Matthew Gormley. 2024. Unlimiformer: Long-range transformers with unlimited length input. *Advances in Neural Information Processing Systems*, 36.

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

Peter Bürgisser, Michael Clausen, and Mohammad A Shokrollahi. 2013. *Algebraic complexity theory*, volume 315. Springer Science & Business Media.

Ruisi Cai, Yuandong Tian, Zhangyang Wang, and Beidi Chen. 2024. Lococo: Dropping in convolutions for long context compression. *arXiv preprint arXiv:2406.05317*.

Yang Cao. 2024. Sorsa: Singular values and orthonormal regularized singular vectors adaptation of large language models. *arXiv preprint arXiv:2409.00055*.

Yang Cao, Bo Chen, Xiaoyu Li, Yingyu Liang, Zhizhou Sha, Zhenmei Shi, Zhao Song, and Mingda Wan. 2025a. Force matching with relativistic constraints: A physics-inspired approach to stable and efficient generative modeling. In *CIKM*.

Yang Cao, Yubin Chen, Zhao Song, and Jiahao Zhang. 2025b. Towards high-order mean flow generative models: Feasibility, expressivity, and provably efficient criteria. *arXiv preprint arXiv:2508.07102*.

Yang Cao, Zhao Song, and Chiwun Yang. 2025c. Video latent flow matching: Optimal polynomial projections for video interpolation and extrapolation. *arXiv preprint arXiv:2502.00500*.

Yuefan Cao, Xuyang Guo, Jiayan Huo, Yingyu Liang, Zhenmei Shi, Zhao Song, Jiahao Zhang, and Zhen Zhuang. 2025d. Text-to-image diffusion models cannot count, and prompt refinement cannot help. *arXiv preprint arXiv:2503.06884*.

Bo Chen, Yingyu Liang, Zhizhou Sha, Zhenmei Shi, and Zhao Song. 2024a. Hsr-enhanced sparse attention acceleration. *arXiv preprint arXiv:2410.10165*.

Lizhang Chen, Jonathan Li, and Qiang Liu. 2025. Muon optimizes under spectral norm constraints. *arXiv preprint arXiv:2506.15054*.

Lizhang Chen, Bo Liu, Kaizhao Liang, and Qiang Liu. 2024b. Lion secretly solves constrained optimization: As lyapunov predicts. *International Conference on Learning Representations, Spotlight*.

Lizhang Chen, Bo Liu, Lemeng Wu, Kaizhao Liang, Jiaxu Zhu, Chen Liang, Raghuraman Krishnamoorthi, and Qiang Liu. 2024c. Communication efficient distributed training with distributed lion. *Advances in Neural Information Processing Systems*, 37:18388–18415.

Sitan Chen, Jerry Li, and Zhao Song. 2020. Learning mixtures of linear regressions in subexponential time via fourier moments. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 587–600.

Xiang Chen, Zhao Song, Baocheng Sun, Junze Yin, and Danyang Zhuo. 2023a. Query complexity of active learning for function family with nearly orthogonal basis. *arXiv preprint arXiv:2306.03356*.

Xue Chen, Daniel M Kane, Eric Price, and Zhao Song. 2016. Fourier-sparse interpolation without a frequency gap. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 741–750. IEEE.

Yubin Chen, Xuyang Guo, Zhenmei Shi, Zhao Song, and Jiahao Zhang. 2026. T2vworldbench: A benchmark for evaluating world knowledge in text-to-video generation. In *WACV*.

Yukang Chen, Shengju Qian, Haotian Tang, Xin Lai, Zhijian Liu, Song Han, and Jiaya Jia. 2023b. Longlora: Efficient fine-tuning of long-context large language models. *arXiv preprint arXiv:2309.12307*.

Lu Chi, Borui Jiang, and Yadong Mu. 2020. Fast fourier convolution. *Advances in Neural Information Processing Systems*, 33:4479–4488.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. 2019. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. 2020. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*.

Martin Courtois, Malte Ostendorff, Leonhard Hennig, and Georg Rehm. 2024. Symmetric dot-product attention for efficient training of bert language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*.

Yichuan Deng, Zhao Song, and Tianyi Zhou. 2023. Superiority of softmax: Unveiling the performance edge over linear attention. *arXiv preprint arXiv:2310.11685*.

Huaian Diao, Zhao Song, Wen Sun, and David Woodruff. 2018. Sketching for kronecker product regression and p-splines. In *International Conference on Artificial Intelligence and Statistics*, pages 1299–1308. PMLR.

Yiran Ding, Li Lyna Zhang, Chengruidong Zhang, Yuanyuan Xu, Ning Shang, Jiahang Xu, Fan Yang, and Mao Yang. 2024. Longrope: Extending llm context window beyond 2 million tokens. *arXiv preprint arXiv:2402.13753*.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, et al. 2021. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 1:1.

Daniel Y Fu, Elliot L Epstein, Eric Nguyen, Armin W Thomas, Michael Zhang, Tri Dao, Atri Rudra, and Christopher Ré. 2023. Simple hardware-efficient long convolutions for sequence modeling. In *International Conference on Machine Learning*, pages 10373–10391. PMLR.

Yeqi Gao, Zhao Song, and Baocheng Sun. 2022. An $O(k \log n)$ time fourier set query algorithm. *arXiv preprint arXiv:2208.09634*.

Yeqi Gao, Zhao Song, Weixin Wang, and Junze Yin. 2023a. A fast optimization view: Reformulating single layer attention in llm based on tensor and svm trick, and solving it in matrix multiplication time. *arXiv preprint arXiv:2309.07418*.

Yeqi Gao, Zhao Song, and Shenghao Xie. 2023b. In-context learning for attention scheme: from single softmax regression to multiple softmax regression via a tensor trick. *arXiv preprint arXiv:2307.02419*.

Yeqi Gao, Zhao Song, and Junze Yin. 2023c. Gradientcoin: A peer-to-peer decentralized large language models. *arXiv preprint arXiv:2308.10502*.

Yeqi Gao, Zhao Song, and Junze Yin. 2023d. An iterative algorithm for rescaled hyperbolic functions regression. *arXiv preprint arXiv:2305.00660*.

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

Albert Gu, Karan Goel, and Christopher Ré. 2021. Efficiently modeling long sequences with structured state spaces. *arXiv preprint arXiv:2111.00396*.

Yuzhou Gu, Zhao Song, Junze Yin, and Lichen Zhang. 2024. Low rank matrix completion via robust alternating minimization in nearly linear time. In *The Twelfth International Conference on Learning Representations*.

Xuyang Guo, Zekai Huang, Jiayan Huo, Yingyu Liang, Zhenmei Shi, Zhao Song, and Jiahao Zhang. 2025a. Can you count to nine? a human evaluation benchmark for counting limits in modern text-to-video models. *arXiv preprint arXiv:2504.04051*.

Xuyang Guo, Jiayan Huo, Zhenmei Shi, Zhao Song, Jiahao Zhang, and Jiale Zhao. 2025b. T2vphysbench: A first-principles benchmark for physical consistency in text-to-video generation. *arXiv preprint arXiv:2505.00337*.

Xuyang Guo, Jiayan Huo, Zhenmei Shi, Zhao Song, Jiahao Zhang, and Jiale Zhao. 2025c. T2vtextbench: A human evaluation benchmark for textual control in video generation models. *arXiv preprint arXiv:2505.04946*.

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

Xinyi Hou, Yanjie Zhao, Yue Liu, Zhou Yang, Kailong Wang, Li Li, Xiapu Luo, David Lo, John Grundy, and Haoyu Wang. 2024. Large language models for software engineering: A systematic literature review. *Preprint*, arXiv:2308.10620.

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

Jerry Yao-Chieh Hu, Bo-Yu Chen, Dennis Wu, Feng Ruan, and Han Liu. 2025a. Nonparametric modern hopfield models. In *Forty-second International Conference on Machine Learning*.

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

Jerry Yao-Chieh Hu, Hude Liu, Hong-Yu Chen, Weimin Wu, and Han Liu. 2025b. Universal approximation with softmax attention. *arXiv preprint arXiv:2504.15956*.

Jerry Yao-Chieh Hu, Hude Liu, Jennifer Yuntong Zhang, and Han Liu. 2025c. In-context algorithm emulation in fixed-weight transformers. *arXiv preprint arXiv:2508.17550*.

Jerry Yao-Chieh Hu, Maojiang Su, En-Jui Kuo, Zhao Song, and Han Liu. 2025d. Computational limits of low-rank adaptation (lora) fine-tuning for transformer models. In *The Thirteenth International Conference on Learning Representations*.

Jerry Yao-Chieh Hu, Wei-Po Wang, Ammar Gilani, Chenyang Li, Zhao Song, and Han Liu. 2025e. Fundamental limits of prompt tuning transformers: Universality, capacity and efficiency. In *The Thirteenth International Conference on Learning Representations*.

Jerry Yao-Chieh Hu, Dennis Wu, and Han Liu. 2024c. Provably optimal memory capacity for modern hopfield models: Tight analysis for transformer-compatible dense associative memories. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37.

Jerry Yao-Chieh Hu, Weimin Wu, Yi-Chen Lee, Yu-Chao Huang, Minshuo Chen, and Han Liu. 2025f. On statistical rates of conditional diffusion transformers: Approximation, estimation and minimax optimality. In *The Thirteenth International Conference on Learning Representations*.

Jerry Yao-Chieh Hu, Weimin Wu, Zhuoru Li, Sophia Pi, , Zhao Song, and Han Liu. 2024d. On statistical rates and provably efficient criteria of latent diffusion transformers (dits). *Advances in Neural Information Processing Systems*, 38.

Jerry Yao-Chieh Hu, Donglin Yang, Dennis Wu, Chenwei Xu, Bo-Yu Chen, and Han Liu. 2023. On sparse modern hopfield model. *Advances in Neural Information Processing Systems*, 37.

Jerry Yao-Chieh Hu, Xiwen Zhang, Maojiang Su, Zhao Song, and Han Liu. 2025g. Minimalist softmax attention provably learns constrained boolean functions. *arXiv preprint arXiv:2505.19531*.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

Hongye Jin, Xiaotian Han, Jingfeng Yang, Zhimeng Jiang, Zirui Liu, Chia-Yuan Chang, Huiyuan Chen, and Xia Hu. 2024. Llm maybe longlm: Self-extend llm context window without tuning. *arXiv preprint arXiv:2401.01325*.

Yaonan Jin, Daogao Liu, and Zhao Song. 2023. Super-resolution and robust sparse continuous fourier transform in any constant dimension: Nearly linear time

and sample complexity. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*.

Enkelejda Kasneci, Kathrin Seßler, Stefan Küchemann, Maria Bannert, Daryna Dementieva, Frank Fischer, Urs Gasser, Georg Groh, Stephan Günnemann, Eyke Hüllermeier, et al. 2023. Chatgpt for good? on opportunities and challenges of large language models for education. *Learning and individual differences*, 103:102274.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. 2020. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International conference on machine learning*, pages 5156–5165. PMLR.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25.

Andrew Lavin and Scott Gray. 2016. Fast algorithms for convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4013–4021.

Yin Tat Lee, Zhao Song, and Qiuyi Zhang. 2019. Solving empirical risk minimization in the current matrix multiplication time. In *COLT*.

Chenyang Li, Yingyu Liang, Zhenmei Shi, and Zhao Song. 2025a. When can we solve the weighted low rank approximation problem in truly subquadratic time? In *International Conference on Artificial Intelligence and Statistics*.

Chenyang Li, Yingyu Liang, Zhenmei Shi, Zhao Song, and Tianyi Zhou. 2024a. Fourier circuits in neural networks and transformers: A case study of modular arithmetic with multiple inputs. *arXiv preprint arXiv:2402.09469*.

Xiaoyu Li, Yingyu Liang, Jiangxuan Long, Zhenmei Shi, Zhao Song, and Zhen Zhuang. 2025b. Neural algorithmic reasoning for hypergraphs with looped transformers. *arXiv preprint arXiv:2501.10688*.

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

Yinheng Li, Shaofei Wang, Han Ding, and Hang Chen. 2023a. Large language models in finance: A survey. In *Proceedings of the Fourth ACM International Conference on AI in Finance*, pages 374–382.

Yuanzhi Li, Yingyu Liang, and Andrej Risteski. 2016. Recovery guarantee of weighted low-rank approximation via alternating minimization. In *International Conference on Machine Learning*, pages 2358–2367. PMLR.

Zhihang Li, Zhao Song, Weixin Wang, Junze Yin, and Zheng Yu. 2024c. How to inverting the leverage score distribution? *arXiv preprint arXiv:2404.13785*.

Zhihang Li, Zhao Song, Zifan Wang, and Junze Yin. 2023b. Local convergence of approximate newton method for two layer nonlinear regression. *arXiv preprint arXiv:2311.15390*.

Kaizhao Liang, Lizhang Chen, Bo Liu, and Qiang Liu. 2024a. Cautious optimizers: Improving training with one line of code. *arXiv preprint arXiv:2411.16085*.

Kaizhao Liang, Bo Liu, Lizhang Chen, and Qiang Liu. 2024b. Memory-efficient llm training with online subspace descent. *Advances in Neural Information Processing Systems*.

Weixin Liang, Zachary Izzo, Yaohui Zhang, Haley Lepp, Hancheng Cao, Xuandong Zhao, Lingjiao Chen, Haotian Ye, Sheng Liu, Zhi Huang, et al. 2024c. Monitoring ai-modified content at scale: A case study on the impact of chatgpt on ai conference peer reviews. *arXiv preprint arXiv:2403.07183*.

Yingyu Liang, Jiangxuan Long, Zhenmei Shi, Zhao Song, and Yufa Zhou. 2024d. Beyond linear approximations: A novel pruning approach for attention matrix. *arXiv preprint arXiv:2410.11261*.

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

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

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

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

Hude Liu, Jerry Yao-Chieh Hu, Zhao Song, and Han Liu. 2025. Attention mechanism, max-affine partition, and universal approximation. *arXiv preprint arXiv:2504.19901*.

Xuezhe Ma, Xiaomeng Yang, Wenhan Xiong, Beidi Chen, Lili Yu, Hao Zhang, Jonathan May, Luke Zettlemoyer, Omer Levy, and Chunting Zhou. 2024. Megalodon: Efficient llm pretraining and inference with unlimited context length. *arXiv preprint arXiv:2404.08801*.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

Stefano Massaroli, Michael Poli, Dan Fu, Hermann Kumbong, Rom Parnichkun, David Romero, Aman Timalsina, Quinn McIntyre, Beidi Chen, Atri Rudra, et al. 2023. Laughing hyena distillery: Extracting compact recurrences from convolutions. *Advances in Neural Information Processing Systems*, 36.

Michael Mathieu, Mikael Henaff, and Yann LeCun. 2013. Fast training of convolutional networks through ffts. *arXiv preprint arXiv:1312.5851*.

Ankur Moitra. 2015. Super-resolution, extremal functions and the condition number of vandermonde matrices. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 821–830.

Son Nguyen, Lizhang Chen, Bo Liu, and Qiang Liu. 2024. H-fac: Memory-efficient optimization with factorized hamiltonian descent. *International Conference on Artificial Intelligence and Statistics*.

Son Nguyen, Bo Liu, Lizhang Chen, and Qiang Liu. 2025. Improving adaptive moment optimization via preconditioner diagonalization. *arXiv preprint arXiv:2502.07488*.

Eshaan Nichani, Alex Damian, and Jason D Lee. 2024. How transformers learn causal structure with gradient descent. *arXiv preprint arXiv:2402.14735*.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, et al. 2022. In-context learning and induction heads. *arXiv preprint arXiv:2209.11895*.

Bo Peng, Eric Alcaide, Quentin Gregory Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, Huanqi Cao, Xin Cheng, Michael Nguyen Chung, Leon Derczynski, et al. 2023. Rwkv: Reinventing rnns for the transformer era. In *The 2023 Conference on Empirical Methods in Natural Language Processing*.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. 2024. Yarn: Efficient context window extension of large language models. In *The Twelfth International Conference on Learning Representations*.

Michael Poli, Stefano Massaroli, Eric Nguyen, Daniel Y Fu, Tri Dao, Stephen Baccus, Yoshua Bengio, Stefano Ermon, and Christopher Ré. 2023. Hyena hierarchy: Towards larger convolutional language models. In *International Conference on Machine Learning*, pages 28043–28078. PMLR.

Harry Pratt, Bryan Williams, Frans Coenen, and Yalin Zheng. 2017. Fcnn: Fourier convolutional neural networks. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2017, Skopje, Macedonia, September 18–22, 2017, Proceedings, Part I 17*, pages 786–798. Springer.

Eric Price and Zhao Song. 2015. A robust sparse Fourier transform in the continuous setting. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 583–600. IEEE.

Zhen Qin, Xiaodong Han, Weixuan Sun, Bowen He, Dong Li, Dongxu Li, Yuchao Dai, Lingpeng Kong, and Yiran Zhong. 2023. Toeplitz neural network for sequence modeling. *arXiv preprint arXiv:2305.04749*.

Ilya Razenshteyn, Zhao Song, and David P Woodruff. 2016. Weighted low rank approximations with provable guarantees. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 250–263.

Aravind Reddy, Zhao Song, and Lichen Zhang. 2022. Dynamic tensor product regression. *Advances in Neural Information Processing Systems*, 35:4791–4804.

Gautam Reddy. 2024. The mechanistic basis of data dependence and abrupt learning in an in-context classification task. In *The Twelfth International Conference on Learning Representations*.

Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. 2021. Linear transformers are secretly fast weight programmers. In *International Conference on Machine Learning*. PMLR.

Florian Sestak, Artur Toshev, Andreas Fürst, Günter Klambauer, Andreas Mayr, and Johannes Brandstetter. 2025. Lam-slide: Latent space modeling of spatial dynamical systems via linked entities. *arXiv preprint arXiv:2502.12128*.

Zhenmei Shi, Jiefeng Chen, Kunyang Li, Jayaram Raghuram, Xi Wu, Yingyu Liang, and Somesh Jha. 2023a. The trade-off between universality and label efficiency of representations from contrastive learning. In *The Eleventh International Conference on Learning Representations*.

Zhenmei Shi, Yifei Ming, Xuan-Phi Nguyen, Yingyu Liang, and Shafiq Joty. 2024. Discovering the gems in early layers: Accelerating long-context llms with 1000x input token reduction. *arXiv preprint arXiv:2409.17422*.

Zhenmei Shi, Junyi Wei, Zhuoyan Xu, and Yingyu Liang. 2023b. Why larger language models do in-context learning differently? In *R0-FoMo:Robustness of Few-shot and Zero-shot Learning in Large Foundation Models*.

Jiajun Song and Yiqiao Zhong. 2023. Uncovering hidden geometry in transformers via disentangling position and context. *arXiv preprint arXiv:2310.04861*.

Zhao Song. 2019. *Matrix Theory: Optimization, Concentration and Algorithms*. Ph.D. thesis, The University of Texas at Austin.

Zhao Song, Baocheng Sun, Omri Weinstein, and Ruizhe Zhang. 2022. Sparse fourier transform over lattices: A unified approach to signal reconstruction. *arXiv preprint arXiv:2205.00658*.

Zhao Song, Baocheng Sun, Omri Weinstein, and Ruizhe Zhang. 2023a. Quartic samples suffice for fourier interpolation. In *2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1414–1425. IEEE.

Zhao Song, Weixin Wang, and Junze Yin. 2023b. A unified scheme of resnet and softmax. *arXiv preprint arXiv:2309.13482*.

Zhao Song, Xin Yang, Yuanyuan Yang, and Lichen Zhang. 2023c. Sketching meets differential privacy: fast algorithm for dynamic kronecker projection maintenance. In *International Conference on Machine Learning*, pages 32418–32462. PMLR.

Zhao Song, Mingquan Ye, Junze Yin, and Lichen Zhang. 2023d. Efficient alternating minimization with applications to weighted low rank approximation. *arXiv preprint arXiv:2306.04169*.

Zhao Song, Mingquan Ye, Junze Yin, and Lichen Zhang. 2023e. A nearly-optimal bound for fast regression with ℓ_∞, guarantee. In *ICML*, volume 202 of *Proceedings of Machine Learning Research*, pages 32463–32482. PMLR.

Zhao Song, Junze Yin, and Lichen Zhang. 2023f. Solving attention kernel regression problem via preconditioner. *arXiv preprint arXiv:2308.14304*.

Zhao Song, Junze Yin, and Lichen Zhang. 2024. Solving attention kernel regression problem via preconditioner. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 208–216. PMLR.

Zhao Song, Lichen Zhang, and Ruizhe Zhang. 2021. Training multi-layer over-parametrized neural network in subquadratic time. *arXiv preprint arXiv:2112.07628*.

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

Yutao Sun, Li Dong, Shaohan Huang, Shuming Ma, Yuqing Xia, Jilong Xue, Jianyong Wang, and Furu Wei. 2023. Retentive network: A successor to transformer for large language models. *arXiv preprint arXiv:2307.08621*.

Zhongxiang Sun. 2023. A short survey of viewing large language models in legal aspect. *arXiv preprint arXiv:2303.09136*.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. 2024. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*.

Arun James Thirunavukarasu, Darren Shu Jeng Ting, Kabilan Elangovan, Laura Gutierrez, Ting Fang Tan, and Daniel Shu Wei Ting. 2023. Large language models in medicine. *Nature medicine*, 29(8):1930–1940.

Yao-Hung Hubert Tsai, Shaojie Bai, Makoto Yamada, Louis-Philippe Morency, and Ruslan Salakhutdinov. 2019. Transformer dissection: a unified understanding of transformer's attention via the lens of kernel. *arXiv preprint arXiv:1908.11775*.

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

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. 2020. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*.

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

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

Weimin Wu, Teng-Yun Hsiao, Jerry Yao-Chieh Hu, Wenxin Zhang, and Han Liu. 2025a. In-context learning as conditioned associative memory retrieval. In *Forty-second International Conference on Machine Learning*.

Weimin Wu, Maojiang Su, Jerry Yao-Chieh Hu, Zhao Song, and Han Liu. 2025b. In-context deep learning via transformer models. In *International Conference on Machine Learning*. PMLR.

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

Zhuoyan Xu, Zhenmei Shi, and Yingyu Liang. 2024b. 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*.

Zhuoyan Xu, Zhenmei Shi, Junyi Wei, Fangzhou Mu, Yin Li, and Yingyu Liang. 2024c. Towards few-shot adaptation of foundation models via multitask finetuning. In *The Twelfth International Conference on Learning Representations*.

Lu Ye, Ze Tao, Yong Huang, and Yang Li. 2024. Chunkattention: Efficient self-attention with prefix-aware kv cache and two-phase partition. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. 2020. Big bird: Transformers for longer sequences. *Advances in neural information processing systems*, 33:17283–17297.

Michael Zhang, Kush Bhatia, Hermann Kumbong, and Christopher Re. 2024. The hedgehog & the porcupine: Expressive linear attentions with softmax mimicry. In *The Twelfth International Conference on Learning Representations*.

Ruiqi Zhang, Spencer Frei, and Peter L Bartlett. 2023. Trained transformers learn linear models in-context. *arXiv preprint arXiv:2306.09927*.

Lin Zheng, Chong Wang, and Lingpeng Kong. 2022. Linear complexity randomized self-attention mechanism. In *International conference on machine learning*, pages 27011–27041. PMLR.

Lei Zhu, Xinjiang Wang, Wayne Zhang, and Rynson WH Lau. 2024. Relayattention for efficient large language model serving with long system prompts. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*.

Appendix

Roadmap. In Section A, we discuss additional related works. In Section B, we present additional details and proofs related to the convolution approximation approach. In Section C, we introduce the conv approximation in gradient. In Section D, we show that our broadly applicable technique extends low-rank approximations of attention matrices and generalizes existing results to more complex settings. In Section E, we include supplementary material for the low-rank approximation. In Section F, we present a collection of useful tools and lemmas that are referenced throughout the main text and the appendix. In Section G, we provide potential risks.

A More Related Work

Fast attention computation and long context LLM. The development of efficient attention computation has been an active area of research in recent years. The standard self-attention mechanism, introduced in the transformer architecture (Vaswani et al., 2017), has a quadratic complexity with respect to the sequence length, which limits its applicability to long sequences. To address this limitation, various approaches have been proposed to improve the efficiency of attention computation. One line of research focuses on patterns of sparse attention that reduce the number of computations (Child et al., 2019; Beltagy et al., 2020; Zaheer et al., 2020; Shi et al., 2023a; Han et al., 2024). Another approach is to use low-rank approximations or random features for the attention matrix (Razenshteyn et al., 2016; Li et al., 2016; Wang et al., 2020; Choromanski et al., 2020; Zheng et al., 2022; Alman and Song, 2023; Ahn et al., 2024; Bebensee and Lee, 2023), which reduces the computational complexity to linear in the sequence length. In addition, using linear attention as a proxy of Softmax attention is a rich line of work (Tsai et al., 2019; Katharopoulos et al., 2020; Schlag et al., 2021; Zhang et al., 2023; Sun et al., 2023; Ahn et al., 2024; Shi et al., 2023b; Xu et al., 2024b; Zhang et al., 2024; Deng et al., 2023). These developments in efficient attention computation have enabled transformer-based models to process longer sequences and have opened up new possibilities for their application in various domains (Chen et al., 2023b; Su et al., 2024; Peng et al., 2024; Ding et al., 2024; Ma et al., 2024; Xu et al., 2024c; An et al., 2024; Bertsch et al., 2024; Chen et al., 2024a; Liang et al., 2024f; Jin et al., 2024; Shi et al., 2024; Ye et al., 2024; Zhu et al., 2024; Courtois et al., 2024). In addition, various efficient attention mechanisms have been developed to reduce the computational cost of attention (Song et al., 2024; Hu et al., 2025b,g; Alman and Song, 2025a,b; Liu et al., 2025; Hu et al., 2025c).

Convolution in language model and FFT. There are many subquadratic-time architectures are proposed to address Transformers' computational inefficiency on long sequences, gated convolution recurrent models (Bai et al., 2018; Fu et al., 2023; Peng et al., 2023; Qin et al., 2023), and structured state space models (SSMs) (Gu et al., 2021; Gu and Dao, 2023). They can use global or local convolution (Krizhevsky et al., 2012) operations to replace attention while keeping a comparable performance. The convolution operation can be computed by fast Fourier transform (FFT) efficiently (Pratt et al., 2017; Chi et al., 2020). Moreover, the development of efficient convolution algorithms like Winograd (Lavin and Gray, 2016) and FFT-based convolutions (Mathieu et al., 2013) has further optimized the computation, reducing the memory footprint and improving the overall speed. There are many other works studying Fourier transform (Price and Song, 2015; Moitra, 2015; Chen et al., 2016; Song, 2019; Lee et al., 2019; Chen et al., 2020; Song et al., 2022; Gao et al., 2022; Song et al., 2023a; Chen et al., 2023a; Song et al., 2023e; Jin et al., 2023).

(Weighted) low rank approximation. Low-rank approximation has become an important tool in machine learning and numerical linear algebra, providing a way to extract the core structure of high-dimensional data while minimizing computational costs. Mathematically, we want to find matrices $X, Y \in \mathbb{R}^{n \times k}$ such that $\|M - XY^\top\|_F$ is minimized. It has been applied to various fields, such as training multi-layer neural network (Song et al., 2021), attention approximation (Alman and Song, 2023, 2024a), dynamic Kronecker product maintenance (Song et al., 2023c), and tensor product regression (Reddy et al., 2022). In practice, certain entries of M tend to be more important than others, leading to the study of the weighted low-rank approximation: finding matrices $X, Y \in \mathbb{R}^{n \times k}$ such that $\|W \circ (M - XY^\top)\|_F$ is minimized, where $W \in \mathbb{R}^{n \times n}_{\geq 0}$ (Li et al., 2016; Razenshteyn et al., 2016; Song et al., 2023d; Gu et al., 2024; Li et al., 2025a). It has been applied to various fields, such

as (Song et al., 2021, 2023c, 2021; Reddy et al., 2022). Weighted low-rank approximation has also been connected to optimization and efficiency in large models. For example, theoretical analyses of Transformer fine-tuning and in-context learning, including LoRA limits (Hu et al., 2025d), prompt tuning capacity and universality (Hu et al., 2025e), and in-context learning as conditioned associative memory retrieval (Wu et al., 2025b,a). As data continues to grow in size and complexity, (weighted) low-rank approximation remains an active area of research, with ongoing efforts to develop more efficient, scalable, and robust methods for a wide range of applications.

Attention optimization. There are several other techniques optimizing the approximation of the attention computation to alleviate the quadratic complexity $O(n^2)$, such as optimizing the attention-related regression problems (Song et al., 2023f; Gao et al., 2023b,d,c; Li et al., 2024b; Liang et al., 2024d), multi-layer attention optimization (Song et al., 2023b; Li et al., 2023b; Liang et al., 2024e), cross attention (Liang et al., 2024g), Hopfield Models (Hu et al., 2023; Wu et al., 2024b; Hu et al., 2024b; Xu et al., 2024a; Wu et al., 2024a; Hu et al., 2024a, 2025a, 2024c), and optimizing the tensor version of the attention approximation (Liang et al., 2024h; Alman and Song, 2024b).

Beyond approximation-based techniques, there is a growing line of theoretical work that investigates optimization in large models, with a particular focus on their fundamental limitations, such as (Chen et al., 2024b; Liang et al., 2024b,a; Cao, 2024; Chen et al., 2024c; Nguyen et al., 2024; Chen et al., 2025; Nguyen et al., 2025; Li et al., 2025b).

Application in Image/Video Generation Diffusion models have emerged as a powerful framework for generative tasks in both image and video domains. Recent works have explored both the theoretical and practical aspects of these models (Hu et al., 2024d, 2025f), as well as a variety of techniques for modeling temporal dynamics and latent representations (Cao et al., 2025c; Sestak et al., 2025; Cao et al., 2025a,b).

Recent work has also focused on addressing challenges such as physical plausibility, world knowledge consistency, and text generation (Guo et al., 2025b; Chen et al., 2026; Guo et al., 2025a; Cao et al., 2025d; Guo et al., 2025c), which are crucial for the practical deployment of generative models in complex scenarios.

B Technical Details About conv Approximation

In Section B.1, we present the background of Toeplitz, circulant, and convolution matrices. In Section B.2, we develop more mathematical tools for studying the conv approximation. In Section B.3, we give the key lemmas we used. In Section B.4, we use these tools and lemmas to prove our main theorem for the conv approximation. In Section B.5, we analyze our case study.

B.1 Properties of Toeplitz, Circulant, and Convolution Matrices

Remark B.1. *The integer i may have different ranges. We will specify these ranges in later text, corresponding to different contexts.*

The Toeplitz matrix is one such structured matrix that has constant values along its diagonals. We define it as follows:

Definition B.2 (Toeplitz matrix)**.** *Given a length-$(2n-1)$ vector $a \in \mathbb{R}^{2n-1}$ (for convenience, we use $a_i \in \mathbb{R}$ to denote the entry of vector where $i \in \{-(n-1), -(n-2), \cdots, 0, \cdots, (n-2), (n-1)\}$), we can formulate a function $\mathsf{Toep} : \mathbb{R}^{2n-1} \to \mathbb{R}^{n \times n}$ as follows*

$$\mathsf{Toep}(a) = \begin{bmatrix} a_0 & a_{-1} & a_{-2} & \cdots & a_{-(n-1)} \\ a_1 & a_0 & a_{-1} & \cdots & a_{-(n-2)} \\ a_2 & a_1 & a_0 & \cdots & a_{-(n-3)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n-1} & a_{n-2} & a_{n-3} & \cdots & a_0 \end{bmatrix}$$

Furthermore, we define the circulant matrix, which is a structured matrix where each row vector is rotated one element to the right relative to the preceding row vector, which is defined as follows:

Definition B.3 (Circulant matrix)**.** *Let $a \in \mathbb{R}^n$ denote a length-n vector. We define $\mathsf{Circ} : \mathbb{R}^n \to \mathbb{R}^{n \times n}$ as,*

$$\mathsf{Circ}(a) := \begin{bmatrix} a_1 & a_n & a_{n-1} & \cdots & a_2 \\ a_2 & a_1 & a_n & \cdots & a_3 \\ a_3 & a_2 & a_1 & \cdots & a_4 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_n & a_{n-1} & a_{n-2} & \cdots & a_1 \end{bmatrix}.$$

Now, we define a binary operation $*$ defined on $\mathbb{R}^d$:

Definition B.4. *Let conv be defined in Definition 2.5. Given two vectors a and $x \in \mathbb{R}^n$, let $a * x \in \mathbb{R}^n$ denote the convolution operator between a and x, i.e., $a * x := \mathsf{conv}(a)x$.*

Finally, we present a basic fact about the Hadamard product.

Fact B.5. *For all $a, b \in \mathbb{R}^n$, we have $a \circ b = b \circ a = \mathrm{diag}(a) \cdot b = \mathrm{diag}(b) \cdot a$.*

Below, we explore the properties of conv, Toep, Resi, and Circ.

Claim B.6. *Given a length-$(2n-1)$ vector $a' \in \mathbb{R}^{2n-1}$ (for convenience, we use $a'_i \in \mathbb{R}$ to denote the entry of vector where $i \in \{-(n-1), -(n-2), \cdots, 0, \cdots, (n-2), (n-1)\}$). Let $a \in \mathbb{R}^n$, such that $a = [a'_0, a'_1, \ldots, a'_{n-1}]^\top$. Let M be defined in Definition 2.2,* Toep *be defined in Definition B.2, and* conv *be defined in Definition 2.5. We have*

$$\mathsf{conv}(a) = \mathsf{Toep}(\begin{bmatrix} \mathbf{0}_{n-1} \\ a \end{bmatrix}) = M \circ \mathsf{Toep}(a').$$

Proof. The proof directly follows the Definition 2.2, Definition B.2, and Definition 2.5. □

Fact B.7 (Folklore)**.** *Let* Toep *be defined in Definition B.2, and* Circ *be defined in Definition B.3. Given a length-$(2n-1)$ vector $a \in \mathbb{R}^{2n-1}$ (for convenience, we use $a_i \in \mathbb{R}$ to denote the entry of vector where $i \in \{-(n-1), -(n-2), \cdots, 0, \cdots, (n-2), (n-1)\}$). Let $a' \in \mathbb{R}^{2n}$, such that $a' = [a_0, a_1, \ldots, a_{n-1}, 0, a_{-(n-1)}, \ldots, a_{-1}]^\top$. For any $x \in \mathbb{R}^n$, we have*

$$\begin{aligned} \mathsf{Circ}(a') \begin{bmatrix} x \\ \mathbf{0}_n \end{bmatrix} &= \begin{bmatrix} \mathsf{Toep}(a) & \mathsf{Resi}(a) \\ \mathsf{Resi}(a) & \mathsf{Toep}(a) \end{bmatrix} \cdot \begin{bmatrix} x \\ \mathbf{0}_n \end{bmatrix} \\ &= \begin{bmatrix} \mathsf{Toep}(a)x \\ \mathsf{Resi}(a)x \end{bmatrix}, \end{aligned}$$

where the residual matrix $\mathsf{Resi}(a)$ *is defined as*

$$\begin{bmatrix} 0 & a_{n-1} & a_{n-2} & \cdots & a_2 & a_1 \\ a_{-(n-1)} & 0 & a_{n-1} & \cdots & a_3 & a_2 \\ a_{-(n-2)} & a_{-(n-1)} & 0 & \cdots & a_4 & a_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{-2} & a_{-3} & a_{-4} & \cdots & 0 & a_{n-1} \\ a_{-1} & a_{-2} & a_{-3} & \cdots & a_{-(n-1)} & 0 \end{bmatrix}.$$

$\mathsf{Circ}(a)$ can be expressed in the form of $F^{-1}\mathrm{diag}(Fa)F$, which is as follows:

Fact B.8 (Folklore)**.** *Let $a \in \mathbb{R}^n$ denote a length-n vector. Let* Circ *be defined in Definition B.3. Let $F \in \mathbb{C}^{n \times n}$ denote the discrete Fourier transform matrix. Using the property of discrete Fourier transform, we have*

$$\mathsf{Circ}(a) = F^{-1}\mathrm{diag}(Fa)F.$$

Claim B.9 (Restatement of Claim 2.6)**.** *We have $\mathsf{conv}(e_j) \in \mathbb{R}^{n \times n}$ is a j-rank matrix, where the j-th entry of $e_j \in \mathbb{R}^n$ is 1 and all other entries are 0.*

Proof. This follows from Definition 2.5. □

Claim B.10 (Restatement of Claim 2.7)**.** *Let* conv *be defined in Definition 2.5. For any $a, x \in \mathbb{R}^n$, $\mathsf{conv}(a)x$ can be computed in $O(n \log n)$ via FFT.*

Proof of Claim 2.7. For any $a \in \mathbb{R}^n$, we denote $a' = \begin{bmatrix} \mathbf{0}_{n-1} \\ a \end{bmatrix} \in \mathbb{R}^{2n-1}$ and $a'' = \begin{bmatrix} a \\ \mathbf{0}_n \end{bmatrix} \in \mathbb{R}^{2n}$. We have

$$\begin{aligned} \begin{bmatrix} \mathsf{conv}(a)x \\ \mathsf{Resi}(a')x \end{bmatrix} &= \begin{bmatrix} \mathsf{Toep}(a')x \\ \mathsf{Resi}(a')x \end{bmatrix} \\ &= \mathsf{Circ}(a'') \begin{bmatrix} x \\ \mathbf{0}_n \end{bmatrix} \\ &= F^{-1}\mathrm{diag}(Fa'')F \begin{bmatrix} x \\ \mathbf{0}_n \end{bmatrix}, \end{aligned}$$

where the first step follows Claim B.6, i.e., $\mathsf{conv}(a) = \mathsf{Toep}(\begin{bmatrix} \mathbf{0}_{n-1} \\ a \end{bmatrix})$, the second step follows Fact B.7 and the last step follows Fact B.8. We finish the proof by $O(n \log n)$ for FFT. □

Claim B.11 (Restatement of Claim 2.8)**.** conv *is additive, i.e., for any $a, b, x \in \mathbb{R}^n$ we have*

$$\mathsf{conv}(a)x + \mathsf{conv}(b)x = \mathsf{conv}(a+b)x.$$

Proof. This follows from Definition 2.5 and the fact that the matrix product operation is additive. □

Claim B.12 (Restatement of Claim 2.10)**.** *Let $m \in [n]$. For any $a, x \in \mathbb{R}^n$, $\mathsf{conv}(a, m)x$, (defined in Definition 2.9) can be computed in $O(n \log n)$ via FFT.*

Proof. This follows from considering the calculation between the truncated matrix of $\mathsf{conv}(a, m)$ and the truncated vector of x with Claim 2.7. □

B.2 Mathematical Tools Development for k-conv Basis

Definition B.13. *Let $M \in \mathbb{R}^{n \times n}$ be defined in Definition 2.2 and $Q, K \in \mathbb{R}^{n \times d}$ be defined in Definition 2.1. We define $\widetilde{H} := M \circ (QK^\top) \in \mathbb{R}^{n \times n}$.*

When a lower triangular matrix H is expressed as the sum of k convolution matrices, it is useful to understand the structure of the entries in H. The following claim provides an explicit formula for the entries of H in terms of the basis vectors of the convolution matrices.

Claim B.14. *Given $b_1, \dots, b_k \in \mathbb{R}^n$ and k integers $m_1, m_2, \dots, m_k$ satisfying $n \geq m_1 > m_2 > \cdots > m_k \geq 1$, let $H = \sum_{i \in [k]} \mathsf{conv}(b_i, m_i)$. Then, for any $i \geq j \in [n]$, let ℓ satisfy $m_\ell \geq n - j + 1$ and $m_{\ell+1} < n - j + 1$, and we have*

$$H_{i,j} = \sum_{l \in [\ell]} (b_l)_{i-j+1}.$$

For any $i < j \in [n]$, we have $H_{i,j} = 0$.

Proof. This is trivial by following $H = \sum_{i \in [k]} \mathsf{conv}(b_i, m_i)$, the Definition 2.5 and Definition 2.9. □

We present the property of $\widetilde{H} = M \circ (QK^\top)$ as follows:

Lemma B.15. *Given $M \in \mathbb{R}^{n \times n}$, $Q, K \in \mathbb{R}^{n \times d}$, and $\widetilde{H} = M \circ (QK^\top)$, we have for any $j \in [n]$, there exists $\widetilde{H}_j \in \mathbb{R}^n$, i.e., the j-th column of $\widetilde{H}$, such that*

$$\widetilde{H}_j = M_j \circ (Q(K^\top)_j)$$

with time complexity $O(nd)$, where $(K^\top)_j$ denotes the j-th row of K.

Proof. We can check the correctness as follows:

$$\begin{aligned} (\widetilde{H})_j &= (M \circ (QK^\top))_j \\ &= M_j \circ (QK^\top)_j \\ &= M_j \circ (Q(K^\top)_j), \end{aligned}$$

where the first step follows from the definition of $\widetilde{H}$ (see Definition B.13), the second step follows from simple algebra, the third step follows from the fact that the j-th column of $K^\top$ is equal to the j-th row of K.

Now, we can check the running time.

- As $Q \in \mathbb{R}^{n \times d}$ and $(K^\top)_j \in \mathbb{R}^d$, we need $O(nd)$ time to get $Q(K^\top)_j$.
- For any vector v, we need $O(n)$ time to get $M_j \circ v$.

Thus, in total, the time complexity is $O(nd)$. □

The key idea behind our approach is to express the matrix exponential of a matrix with k-conv basis as the sum of k sub-convolution matrices involving the basis vectors. This allows us to efficiently approximate the exponential of the attention matrix. We show how to compute the new basis vectors of the convolution matrices from the original basis vectors below.

Lemma B.16. *Let M be a mask defined in Definition 2.2. Given $b_1, \dots, b_k \in \mathbb{R}^n$ and k integers $m_1, m_2, \dots, m_k$ satisfying $n \geq m_1 > m_2 > \cdots > m_k \geq 1$, we let $H = \sum_{r \in [k]} \mathsf{conv}(b_r, m_r)$. We denote $\widetilde{b}_1 = \exp(b_1)$. Then, we can get $\widetilde{b}_2, \widetilde{b}_3, \dots \widetilde{b}_k \in \mathbb{R}^n$ such that for any $r \in \{2, 3, \cdots, k\}$*

$$\widetilde{b}_r = \exp(\sum_{l \in [r]} b_l) - \exp(\sum_{l \in [r-1]} b_l)$$

and $M \circ \exp(H) = \sum_{r \in [k]} \mathsf{conv}(\widetilde{b}_r, m_r)$ with time complexity $O(nk)$.

Proof. **Correctness.**

By Claim B.14, for any $i \geq j \in [n]$, let ℓ satisfy $m_\ell \geq n - j + 1$ and $m_{\ell+1} < n - j + 1$, and we have

$$H_{i,j} = \sum_{l \in [\ell]} (b_l)_{i-j+1}. \tag{1}$$

As exp is an element-wise function, when $i \geq j$ we have $(M \circ \exp(H))_{i,j} = \exp(H)_{i,j}$ and

$$\begin{aligned} & \exp(H)_{i,j} \\ = & \exp(\sum_{l \in [\ell]} (b_l)_{i-j+1}) \\ = & \sum_{r=1}^{\ell} \exp(\sum_{l \in [r]} (b_l)_{i-j+1}) - \exp(\sum_{l \in [r-1]} (b_l)_{i-j+1}) \\ = & \sum_{r=1}^{\ell} (\widetilde{b}_r)_{i-j+1} \\ = & \sum_{r=1}^{\ell} \mathsf{conv}(\widetilde{b}_r, m_r)_{i,j} \\ = & \sum_{r=1}^{k} \mathsf{conv}(\widetilde{b}_r, m_r)_{i,j}, \end{aligned}$$

where the first step follows from Eq. (1), the second step follows from simple algebra, the third step follows from the lemma statement, the fourth step follows from Definition 2.9, and the last step

follows from Definition 2.9 (when $k < r \leq \ell$, $\mathsf{conv}(\widetilde{b}_r, m_r)_{i,j} = 0$).

When $i < j$ we have $(M \circ \exp(H))_{i,j} = 0 = \sum_{r=1}^{k} \mathsf{conv}(\widetilde{b}_r, m_r)_{i,j}$.

Thus, we have $M \circ \exp(H) = \sum_{r \in [k]} \mathsf{conv}(\widetilde{b}_r, m_r)$.

Running time.

We need $O(nk)$ time to get $\sum_{l \in [r]} b_l$ for any $r \in [k]$. Then, we need $O(1)$ time for element-wise exp and minus operation for $O(nk)$ terms. Thus, in total, we need $O(nk)$ time complexity. □

Lemma B.17. *Let $G \in \mathbb{R}^{n \times n}$. Let $M \in \{0,1\}^{n \times n}$. Let $H = M \circ G$ and $A = M \circ \exp(G)$. Then, we have*

$$A = M \circ \exp(H).$$

Proof. We have

$$\begin{aligned} A &= M \circ \exp(G) \\ &= M \circ \exp(M \circ G) \\ &= M \circ \exp(H), \end{aligned}$$

where the first step follows the lemma statement, the second step follows the property of Hadamard product and the last step follows the lemma statement. □

Theorem B.18 (Restatement of Theorem 3.3)**.** *For any lower triangular matrix $G \neq \mathbf{0}_{n \times n} \in \mathbb{R}^{n \times n}$, there exists $k, T \in [n]$ and $\delta, \epsilon \geq 0$ such that G is a ϵ-close (T, δ)-non-degenerate k-conv basis matrix.*

Proof. By Lemma 2.12, we have G is a matrix with k-conv basis for some $k \in [n]$. We finish the proof by setting $T = 1$ and $\delta = \epsilon = 0$. □

B.3 Lemma Used in Main Theorem Proof

In this section, we present the formal proof for our conv approximation main result. In Algorithm 2, we recover the k-conv basis vectors $b'_1, \dots, b'_k \in \mathbb{R}^n$ through an iterative process. We show that after each iteration i, the algorithm maintains certain invariants related to the recovered basis vectors $b'_1, \dots, b'_i \in \mathbb{R}^n$, the index s, and the error compared to the true basis vectors $b_1, \dots, b_i \in \mathbb{R}^n$. These properties allow us to prove the correctness of the overall algorithm. The following lemma formalizes these invariants:

Lemma B.19. *Let $\widetilde{H}$ be a ϵ-close (T, δ)-non-degenerate k-conv basis matrix as defined in Definition 3.2, where $\delta, \epsilon \geq 0$ and $k, T \in [n]$. Let $Q, K, V \in \mathbb{R}^{n \times d}$. In Algorithm 2, we can get $b'_1, \dots, b'_k \in \mathbb{R}^n$. Then, for any $i \in [k]$, after the i-th loop, we have*

- *Part 1: $v = \sum_{r \in [i]} (b'_r)_{1:T}$ and $u = \sum_{r \in [i]} b'_r$*
- *Part 2: $s = n - m_i + 1$*
- *Part 3: $\| \sum_{r \in [i]} (b'_r)_{1:T} - \sum_{r \in [i]} (b_r)_{1:T} \|_1 \leq T\epsilon$*
- *Part 4: $| \sum_{r \in [i]} (b'_r)_l - \sum_{r \in [i]} (b_r)_l | \leq \epsilon$ for any $l \in [n]$.*

Proof. We use the math induction to prove the correctness.

Let $b'_1, \dots, b'_k \in \mathbb{R}^n$ and $v \in \mathbb{R}^T$ defined in Algorithm 2. Let $i \in \{0, \dots, k-1\}$ be fixed. Suppose after the i-th loop, we have

- Part 1: $v = \sum_{r \in [i]} (b'_r)_{1:T}$ and $u = \sum_{r \in [i]} b'_r$
- Part 2: $s = n - m_i + 1$ (Denote $s = 0$, after the 0-th loop.)
- Part 3: $\| \sum_{r \in [i]} (b'_r)_{1:T} - \sum_{r \in [i]} (b_r)_{1:T} \|_1 \leq T\epsilon$
- Part 4: $| \sum_{r \in [i]} (b'_r)_l - \sum_{r \in [i]} (b_r)_l | \leq \epsilon$ for any $l \in [n]$

Now we consider after the $i+1$-th loop.

Proof of Part 1.

We have $v = \sum_{r \in [i+1]} (b'_r)_{1:T}$ and $u = \sum_{r \in [i+1]} b'_r$ by the line 9 and line 10 in Algorithm 2.

Proof of Part 2.

We denote the output of $\textsc{Search}(Q, K, k, T, \delta, \epsilon, \sum_{r \in [i]} (b'_r)_{1:T}, m_i, n - T + 1)$ as y. Now, we prove $y = n - m_{i+1} + 1$.

It is clear that $n - m_i + 1 \leq y \leq n - T + 1$. For any $j \in \{n - m_i + 1, \dots, n - T + 1\}$, we have line 7 in Algorithm 3 as

$$\begin{aligned} \alpha &= \|(\widetilde{H}_j)_{j:j+T-1} - v\|_1 \\ &= \|(H_j)_{j:j+T-1} + R_{j,j:j+T-1} - v\|_1 \\ &= \|(H_j)_{j:j+T-1} + R_{j,j:j+T-1} - \sum_{r \in [i]} (b'_r)_{1:T}\|_1 \\ &= \|(\sum_{r \in [k]} \mathsf{conv}(b_r, m_r))_{j,j:j+T-1} \\ &\quad + R_{j,j:j+T-1} - \sum_{r \in [i]} (b'_r)_{1:T}\|_1, \end{aligned} \tag{2}$$

where the first step follows from Definition B.13 ($\widetilde{H} = H + R$), the second step follows from Part

1, and the last step follows from Definition 3.2 ($H = \sum_{r\in[k]} \mathsf{conv}(b_r, m_r)$).

When $j < n - m_{i+1} + 1$, we have Eq. (2) as

$$\begin{aligned}
&\|(\sum_{r\in[k]} \mathsf{conv}(b_r, m_r))_{j,j:j+T-1} \\
&\quad + R_{j,j:j+T-1} - \sum_{r\in[i]} (b'_r)_{1:T}\|_1 \\
&\le \|(\sum_{r\in[k]} \mathsf{conv}(b_r, m_r))_{j,j:j+T-1} \\
&\quad - \sum_{r\in[i]} (b'_r)_{1:T}\|_1 + \|R_{j,j:j+T-1}\|_1 \\
&\le \|(\sum_{r\in[k]} \mathsf{conv}(b_r, m_r))_{j,j:j+T-1} \\
&\quad - \sum_{r\in[i]} (b'_r)_{1:T}\|_1 + T\epsilon \\
&= \|(\sum_{r\in[i]} \mathsf{conv}(b_r, m_r))_{j,j:j+T-1} \\
&\quad - \sum_{r\in[i]} (b'_r)_{1:T}\|_1 + T\epsilon \\
&= \|\sum_{r\in[i]} (b_r)_{1:T} - \sum_{r\in[i]} (b'_r)_{1:T}\|_1 + T\epsilon \\
&\le 2T\epsilon \\
&< \delta - 2T\epsilon,
\end{aligned}$$

where the first step follows from the triangle inequality, the second step follows from Definition 3.2 ($\|R\|_\infty \le \epsilon$), the third step follows from $j < n - m_{i+1} + 1$, the fourth step follows from Definition 2.9, the fifth step follows from Part 3, and the last step follows from Definition 3.2 ($\epsilon \le \frac{\delta}{5T} < \frac{\delta}{4T}$).

Similarly, when $j \ge n - m_{i+1} + 1$, we have Eq. (2) as

$$\begin{aligned}
&\|(\sum_{r\in[k]} \mathsf{conv}(b_r, m_r))_{j,j:j+T-1} + R_{j,j:j+T-1} \\
&\quad - \sum_{r\in[i]} (b'_r)_{1:T}\|_1 \\
&\ge \|(\sum_{r\in[k]} \mathsf{conv}(b_r, m_r))_{j,j:j+T-1} \\
&\quad - \sum_{r\in[i]} (b'_r)_{1:T}\|_1 - \|R_{j,j:j+T-1}\|_1 \\
&\ge \|(\sum_{r\in[k]} \mathsf{conv}(b_r, m_r))_{j,j:j+T-1} - \sum_{r\in[i]} (b'_r)_{1:T}\|_1 \\
&\quad - T\epsilon \\
&= \|(\sum_{r\in[k]} \mathsf{conv}(b_r, m_r))_{j,j:j+T-1} - \sum_{r\in[i]} (b_r)_{1:T} \\
&\quad + \sum_{r\in[i]} (b_r)_{1:T} - \sum_{r\in[i]} (b'_r)_{1:T}\|_1 - T\epsilon \\
&\ge \|(\sum_{r\in[k]} \mathsf{conv}(b_r, m_r))_{j,j:j+T-1} - \sum_{r\in[i]} (b_r)_{1:T}\|_1 \\
&\quad - \|\sum_{r\in[i]} (b_r)_{1:T} - \sum_{r\in[i]} (b'_r)_{1:T}\|_1 - T\epsilon \\
&\ge \|(\sum_{r\in[k]} \mathsf{conv}(b_r, m_r))_{j,j:j+T-1} - \sum_{r\in[i]} (b_r)_{1:T}\|_1 \\
&\quad - 2T\epsilon \\
&\ge \delta - 2T\epsilon
\end{aligned}$$

where the first step follows from the triangle inequality, the second step follows from Definition 3.2 ($\|R\|_\infty \le \epsilon$), the third step follows from simple algebra, the fourth step follows from the triangle inequality, the fifth step follows from Part 3, and the last step follows from Definition 3.1.

Thus, we can claim, when $\alpha < \delta - 2T\epsilon$, we have $j < n - m_{i+1} + 1$, and we have $j \ge n - m_{i+1} + 1$ otherwise. Therefore, by binary search, we can get $s = y = n - m_{i+1} + 1$.

Proof of Part 3.

We have $s = n - m_{i+1} + 1$ and $u = \sum_{r\in[i]} b'_r$ at line 8 in Algorithm 2. Thus, we have

$$\begin{aligned}
&\|\sum_{r\in[i+1]} (b'_r)_{1:T} - \sum_{r\in[i+1]} (b_r)_{1:T}\|_1 \\
&= \|(b'_{i+1})_{1:T} + \sum_{r\in[i]} (b'_r)_{1:T} - \sum_{r\in[i+1]} (b_r)_{1:T}\|_1 \\
&= \|\widetilde{H}_{s,s:s+T-1} - u_{1:T} + \sum_{r\in[i]} (b'_r)_{1:T} \\
&\quad - \sum_{r\in[i+1]} (b_r)_{1:T}\|_1 \\
&= \|\widetilde{H}_{s,s:s+T-1} - \sum_{r\in[i+1]} (b_r)_{1:T}\|_1 \\
&= \|H_{s,s:s+T-1} + R_{s,s:s+T-1} - \sum_{r\in[i+1]} (b_r)_{1:T}\|_1 \\
&= \|\sum_{r\in[k]} \mathsf{conv}(b_r, m_r)_{s,s:s+T-1} + R_{s,s:s+T-1} \\
&\quad - \sum_{r\in[i+1]} (b_r)_{1:T}\|_1 \\
&= \|\sum_{r\in[i+1]} \mathsf{conv}(b_r, m_r)_{s,s:s+T-1} + R_{s,s:s+T-1} \\
&\quad - \sum_{r\in[i+1]} (b_r)_{1:T}\|_1 \\
&= \|\sum_{r\in[i+1]} (b_r)_{1:T} + R_{s,s:s+T-1} - \sum_{r\in[i+1]} (b_r)_{1:T}\|_1
\end{aligned}$$

$= \|R_{s,s:s+T-1}\|_1$
$\leq T\epsilon$,

where the first step follows from simple algebra, the second step follows from Algorithm 2 (line 8), the third step follows from $u = \sum_{r\in[i]} b'_r$, the fourth step follows from Definition B.13 ($\widetilde{H} = H + R$), the fifth step follows from Definition 3.2 ($H = \sum_{r\in[k]} \mathsf{conv}(b_r, m_r)$), the sixth step follows from $s = n - m_{i+1} + 1$, the seventh step follows from Definition 2.9, the eighth step follows from simple algebra, and the last step follows from Definition 3.2 ($\|R\|_\infty \leq \epsilon$).

Proof of Part 4.

We can get $|\sum_{r\in[i+1]}(b'_r)_l - \sum_{r\in[i]}(b_r)_l| \leq \epsilon$ for any $l \in [n]$ similarly as Proof of Part 3.

We can check the initial conditions hold. Thus, we finish the whole proof by math induction. □

Building upon Lemma B.19, we now analyze the overall error of our approach for approximating the attention computation. Recall that our goal is to efficiently approximate the matrix $Y = D^{-1}AV$, where $A = M \circ \exp(QK^\top)$ and $D = \mathrm{diag}(A\mathbf{1}_n)$. We will show that by using the approximate basis vectors recovered by Algorithm 2, we can construct matrices $\widetilde{A}$ and $\widetilde{D}$ such that the approximation error $\|Y - \widetilde{D}^{-1}\widetilde{A}V\|_\infty$ is bounded. The following lemma provides this error analysis:

Lemma B.20 (Error analysis)**.** *Let $\widetilde{H}$ be a ϵ-close (T,δ)-non-degenerate k-*conv *basis matrix as defined in Definition 3.2, where $\delta, \epsilon \geq 0$ and $k, T \in [n]$. Let $Q, K, V \in \mathbb{R}^{n\times d}$. Recall $A = M \circ \exp(QK^\top)$ and $D = \mathrm{diag}(A\mathbf{1}_n)$ defined in Definition 2.3. By Algorithm 2, we can get k-*conv *basis $\widetilde{b}_1, \ldots, \widetilde{b}_k \in \mathbb{R}^n$ and k integers $m_1, m_2, \ldots, m_k$ satisfying $n \geq m_1 > m_2 > \cdots > m_k \geq T$, such that $\widetilde{A} := \sum_{r\in[k]} \mathsf{conv}(\widetilde{b}_r, m_r)$ and $\widetilde{D} := \mathrm{diag}(\widetilde{A}\mathbf{1}_n)$ satisfy*

$$\|D^{-1}AV - \widetilde{D}^{-1}\widetilde{A}V\|_\infty \leq 2(\exp(2\epsilon) - 1)\|V\|_\infty,$$

with time complexity $O(knd\log(n))$.

Proof. **Correctness.**

By Lemma B.19 Part 4, we can get $b'_1, \ldots, b'_k \in \mathbb{R}^n$, such that, for any $i \in [k]$ and $l \in [n]$, we have

$$|\sum_{r\in[i]}(b'_r)_l - \sum_{r\in[i]}(b_r)_l| \leq \epsilon. \tag{3}$$

Furthermore, we denote

$$H' = \sum_{r\in[k]} \mathsf{conv}(b'_r, m_r)$$

Recall $\widetilde{H} = H + R \in \mathbb{R}^{n\times n}$,

$$H = \sum_{r\in[k]} \mathsf{conv}(b_r, m_r),$$

and $\|R\|_\infty \leq \epsilon$.

Thus, we have

$$\begin{aligned}\|H' - \widetilde{H}\|_\infty &\leq \|H' - H\|_\infty + \|H - \widetilde{H}\|_\infty \\ &\leq \|H' - H\|_\infty + \|R\|_\infty \\ &\leq 2\epsilon, \end{aligned} \tag{4}$$

where the first step follows from triangle inequality, the second step follows from $\widetilde{H} = H + R$ and the last step follows from $\|R\|_\infty \leq \epsilon$ and Eq. (3).

By Lemma B.17, we have

$$\begin{aligned} A &= M \circ \exp(QK^\top) \\ &= M \circ \exp(M \circ QK^\top) \\ &= M \circ \exp(\widetilde{H}). \end{aligned}$$

We also have

$$\widetilde{A} = \sum_{r\in[k]} \mathsf{conv}(\widetilde{b}_r, m_r) = M \circ \exp(H')$$

by Lemma B.16 and line 12 in Algorithm 2.

Then, by Lemma F.4, we have

$$\|D^{-1}AV - \widetilde{D}^{-1}\widetilde{A}V\|_\infty \leq 2(\exp(2\epsilon) - 1)\|V\|_\infty.$$

Running time.

We have k loops in Algorithm 2.

In each loop, we call $O(\log(n))$ times of binary search function. In each binary search function, we take $O(nd)$ time for line 6 in Algorithm 3 by Lemma B.15. Thus, we take $O(nd\log(n))$ in total for the search (Algorithm 3) in each loop.

In each loop, we take $O(nd)$ time for line 7 in Algorithm 2 by Lemma B.15.

Thus, we take total $O(k(nd + nd\log(n))) = O(knd\log(n))$ for the whole loop.

We take $O(nk)$ time for the line 12 in Algorithm 2 by Lemma B.16.

In total, we take $O(nk + knd\log(n)) = O(knd\log(n))$ time. □

We are now ready to prove our main result for the conv approximation approach. Theorem B.21 brings together the key components we have developed: the existence of a k-conv basis for the attention matrix (Definition 3.2), the ability to efficiently recover an approximate k-conv basis (Algorithm 2 and Lemma B.19), and the bounded approximation error when using this approximate basis

(Lemma B.20). The theorem statement is a formal version of our main conv result, Theorem 3.4 and Algorithm 1, which was presented in the main text. It specifies the input properties, the approximation guarantees, and the time complexity of our approach.

B.4 Proof of Main Theorem

Theorem B.21 (Main conv results for inference (Restatement of Theorem 3.4))**.** *Let $Q, K, V \in \mathbb{R}^{n \times d}$. Recall $A = M \circ \exp(QK^\top) \in \mathbb{R}^{n \times n}$, $D = \mathrm{diag}(A\mathbf{1}_n) \in \mathbb{R}^{n \times n}$ defined in Definition 2.3. We denote $Y := D^{-1}AV \in \mathbb{R}^{n \times d}$. Let $M \circ (QK^\top)$ be a ϵ-close (T, δ)-non-degenerate k-conv basis matrix as defined in Definition 3.2, where $\delta, \epsilon \geq 0$ and $k, T \in [n]$. By Algorithm 1, we can get $\widetilde{Y}$ such that*

$$\|Y - \widetilde{Y}\|_\infty \leq 2(\exp(2\epsilon) - 1)\|V\|_\infty,$$

whose time complexity is $O(knd \log(n))$ given M, Q, K, V.

Proof of Theorem 3.4. **Correctness.**

Correctness follows Lemma B.20.

Running time.

By Lemma B.20, we need time $O(knd \log(n))$ time to get k-conv basis $\widetilde{b}_1, \ldots, \widetilde{b}_k \in \mathbb{R}^n$ and k integers $m_1, m_2, \ldots, m_k$ satisfying $n \geq m_1 > m_2 > \cdots > m_k \geq T$.

Denote $\widetilde{A} := \sum_{r \in [k]} \mathsf{conv}(\widetilde{b}_r, m_r)$. By Claim 2.10, we take $O(knd \log(n))$ time to get $\widetilde{A}V$ via FFT as k-conv basis and d columns in V. Similarly, by Claim 2.10, we take $O(kn \log(n))$ time for $\widetilde{D} = \mathrm{diag}(\widetilde{A}\mathbf{1}_n)$ via FFT as k-conv basis. Finally, we take $O(nd)$ time to get $\widetilde{D}^{-1}\widetilde{A}V$ as $\widetilde{D}^{-1}$ is a diagonal matrix.

Thus, in total, we take $O(knd \log(n) + knd \log(n) + kn \log(n) + nd) = O(knd \log(n))$ time complexity. □

Corollary B.22 (Exact conv inference, restatement of Corollary 3.5)**.** *Let $Q, K, V \in \mathbb{R}^{n \times d}$. Recall $A = M \circ \exp(QK^\top) \in \mathbb{R}^{n \times n}$, $D = \mathrm{diag}(A\mathbf{1}_n) \in \mathbb{R}^{n \times n}$ defined in Definition 2.3. We denote $Y := D^{-1}AV \in \mathbb{R}^{n \times d}$. For any $\epsilon \geq 0$ and any Q, K, V, there exists hyper-parameter $k, T \in [n]$ and $\delta \geq 0$ such that Algorithm 1 can output $\widetilde{Y}$ satisfying*

$$\|Y - \widetilde{Y}\|_\infty \leq 2(\exp(2\epsilon) - 1)\|V\|_\infty.$$

Furthermore, we can exactly get Y, i.e., $\epsilon = 0$, through Algorithm 1 with time complexity $O(n^2 d \log(n))$ in the worst case.

Proof. We set $k = n$, $T = 1$, $\delta = 0$ and $\epsilon = 0$ as the input of Algorithm 1. Then, the proof follows Theorem 3.3 and Theorem 3.4 . □

B.5 Construction for Case Study

In this section, we present the case study. We use $\mathbf{i}$ to denote the $\sqrt{-1}$. For a complex number $z = a + b\mathbf{i} \in \mathbb{C}$, where $a, b \in \mathbb{R}$, we use $|z|$ to denote its norm, i.e., $|z| = \sqrt{a^2 + b^2}$.

Lemma B.23 (Complex vector construction)**.** *If the vectors $x_1, \cdots, x_n \in \mathbb{C}^d$ satisfy the following properties,*

- $\|x_i\|_2 = 1$ *for all* $i \in [n]$
- *For each $i \in [n]$, let $x_{i,1} = e^{\mathbf{i}i\theta}$ and $e_{i,l} = 0$ for all $l \neq 1$*

Then we have for all $i \in [n]$, for all $j \in [n]$, $\|x_i - x_j\|_2^2 = f(i - j)$ for some function f.

Proof. We can show that

$$\begin{aligned} \|x_i - x_j\|_2^2 &= |e^{\mathbf{i}i\theta} - e^{\mathbf{i}j\theta}|^2 \\ &= |e^{\mathbf{i}j\theta}|^2 \cdot |e^{\mathbf{i}(i-j)\theta} - 1|^2 \\ &= |e^{\mathbf{i}(i-j)\theta} - 1|^2 \\ &=: f(i - j), \end{aligned}$$

where the first step follows from the assumption that for each $i \in [n]$ and $l \neq 1$, $x_{i,1} = e^{\mathbf{i}i\theta}$ and $e_{i,l} = 0$, the second step follows from simple algebra, the third step follows from the $|e^{\mathbf{i}j\theta}| = 1$, and the last step follows from the definition of the function f.

Thus, we complete the proof. □

Lemma B.24 (Real vector construction)**.** *If the vectors $x_1, \cdots, x_n \in \mathbb{R}^d$ satisfy the following properties,*

- $\|x_i\|_2 = 1$ *for all* $i \in [n]$
- $x_{i,1} = \cos(i\theta)$ *and* $x_{i,2} = \sin(i\theta)$. *For all $l \notin \{1, 2\}$, we have $x_{i,l} = 0$.*

Then we have for all $i \in [n]$, for all $j \in [n]$, $\|x_i - x_j\|_2^2 = f(i - j)$ for some function f.

Proof. We can show that

$$\begin{aligned} &\|x_i - x_j\|_2^2 \\ =\ &(\cos(i\theta) - \cos(j\theta))^2 + (\sin(i\theta) - \sin(j\theta))^2 \\ =\ &2 - 2\cos(i\theta)\cos(j\theta) - 2\sin(i\theta)\sin(j\theta) \\ =\ &2 - 2\cos((i - j)\theta), \end{aligned}$$

where the first step follows from construction condition, the second step follows from simple algebra, and the last step follows from the trigonometric properties.

Thus, we complete the proof. □

Lemma B.25 (A general real vector construction)**.** *If the vectors $x_1, \cdots, x_n \in \mathbb{R}^d$ satisfy the following properties,*

- $\|x_i\|_2 = 1$ *for all* $i \in [n]$.
- *Let* $H \in \mathbb{R}^{d \times d}$ *be any orthonormal matrix.*
- *Let* $(s_1, s_2, \ldots, s_d)$ *be a permutation of* $(1, 2, \ldots, d)$.
- *Let* $l = \lfloor (d+1)/2 \rfloor$*, where* l *is an integer. Let* $a_1, \ldots, a_l \in \mathbb{R}$.
- *Let* $u_1, \cdots, u_n \in \mathbb{R}^d$ *and* $x_i = H u_i$ *for any* $i \in [n]$.
- *When* d *is even,* $u_{i,s_k} = a_k \cos(i\theta_k)$ *and* $u_{i,s_{k+l}} = a_k \sin(i\theta_k)$*, for all* $k \in [l]$ *and* $i \in [n]$*, where* $\theta_1, \ldots, \theta_l \in \mathbb{R}$.
- *When* d *is odd,* $u_{i,s_k} = a_k \cos(i\theta_k)$ *and* $u_{i,s_{k+l}} = a_k \sin(i\theta_k)$*, for all* $k \in [l-1]$ *and* $i \in [n]$*, where* $\theta_1, \ldots, \theta_{l-1} \in \mathbb{R}$*, and* $u_{i,s_l} = a_l$.

Then we have for all $i \in [n]$*, for all* $j \in [n]$*,* $\|x_i - x_j\|_2^2 = f(i-j)$ *for some function* f.

Proof. When d is even, we can show that

$$\begin{aligned}
& \|x_i - x_j\|_2^2 \\
= & \|u_i - u_j\|_2^2 \\
= & \sum_{k \in [l]} (a_k \cos(i\theta_k) - a_k \cos(j\theta_k))^2 \\
& + (a_k \sin(i\theta_k) - a_k \sin(j\theta_k))^2 \\
= & \sum_{k \in [l]} a_k^2 \cos^2(i\theta_k) + a_k^2 \cos^2(j\theta_k) \\
& - 2a_k^2 \cos(i\theta_k)\cos(j\theta_k) \\
& + a_k^2 \sin^2(i\theta_k) + a_k^2 \sin^2(j\theta_k) \\
& - 2a_k^2 \sin(i\theta_k)\sin(j\theta_k) \\
= & \sum_{k \in [l]} 2a_k^2 - 2a_k^2 \cos(i\theta_k)\cos(j\theta_k) \\
& - 2a_k^2 \sin(i\theta_k)\sin(j\theta_k) \\
= & \sum_{k \in [l]} 2a_k^2 (1 - \cos(i\theta_k)\cos(j\theta_k) \\
& - \sin(i\theta_k)\sin(j\theta_k)) \\
= & \sum_{k \in [l]} 2a_k^2 (1 - \cos((i-j)\theta_k)),
\end{aligned}$$

where the first step follows H being orthonormal, which preserves the Euclidean distance between two vectors, i.e., $\|Hu_1 - Hu_2\|_2 = \|u_1 - u_2\|_2$ for any $u_1, u_2 \in \mathbb{R}^d$, the second step follows from the construction condition, the third step follows from $(a-b)^2 = a^2 + b^2 - 2ab$ for all $a, b \in \mathbb{C}$, the fourth step follows from $\sin^2(x) + \cos^2(x) = 1$, the fifth step follows from simple algebra, and the last step follows from the trigonometric properties.

When d is odd, we can show similar results by the same way. Thus, we complete the proof. □

Lemma B.26. *If the following conditions hold*

- *Let* $b \in \mathbb{R}^n$ *denote a vector*
- $Q \in \mathbb{R}^{n \times d}$ *and* $K \in \mathbb{R}^{n \times d}$
- *For each* $i, j \in [n]$*,*
  - $(QK^\top)_{i,j} = b_{i-j+1}$ *if* $i \geq j$
  - $(QK^\top)_{i,j} = b_{i-j+n+1}$ *if* $i < j$

Then, there is a vector $a = \exp(b)$ *such that*

$$\exp(QK^\top) = \mathsf{Circ}(a)$$

Proof. Since $a = \exp(b)$, we have

$$\begin{aligned}
\mathsf{Circ}(a) &= \mathsf{Circ}(\exp(b)) \\
&= \exp(\mathsf{Circ}(b)), \qquad (5)
\end{aligned}$$

where the second step follows from the fact that $\exp(\cdot)$ is applied entry-wisely to a vector.

By the assumption from the Lemma statement that $(QK^\top)_{i,j} = b_{i-j+1}$ if $i \geq j$ and $(QK^\top)_{i,j} = b_{i-j+n+1}$ if $i < j$, we get

$$QK^\top = \begin{bmatrix} b_1 & b_n & b_{n-1} & \cdots & b_2 \\ b_2 & b_1 & b_n & \cdots & b_3 \\ b_3 & b_2 & b_1 & \cdots & b_4 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ b_n & b_{n-1} & b_{n-2} & \cdots & b_1 \end{bmatrix},$$

which is exactly equal to $\mathsf{Circ}(b)$ (see Definition B.3).

Therefore, combining with Eq. (5), we have

$$\exp(QK^\top) = \mathsf{Circ}(a),$$

which completes the proof. □

Lemma B.27. *If the following conditions hold*

- *Let $b \in \mathbb{R}^{2n-1}$ denote a vector*
- *$Q \in \mathbb{R}^{n \times d}$ and $K \in \mathbb{R}^{n \times d}$*
- *For each $i, j \in [n]$, $(QK^\top)_{i,j} = b_{i-j}$.*

Then, there is a vector $a = \exp(b)$ such that

$$\exp(QK^\top) = \mathsf{Toep}(a).$$

Proof. We can prove similarly as Lemma B.26. □

Assumption B.28. *We assume that $W_Q W_K^\top$ is a p.s.d. matrix, so that $W_Q W_K^\top = AA^\top$ where $A \in \mathbb{R}^{d \times d}$.*

Definition B.29. *Assume Assumption B.28. We define $Z := XA \in \mathbb{R}^{n \times d}$, where $Z = \begin{bmatrix} z_1^\top \\ \vdots \\ z_n^\top \end{bmatrix}$.*

Then we have $QK^\top = ZZ^\top$.

Lemma B.30. *If the following conditions hold,*

- *Assume Assumption B.28.*
- *Let $b \in \mathbb{R}^{2n-1}$ denote a vector*
- *Let $z_1, \ldots, z_n$ defined in Definition B.29 satisfy the properties in Lemma B.25.*

Then, there is a vector $a = \exp(b)$ such that

$$\exp(QK^\top) = \mathsf{Toep}(a).$$

Proof. By Lemma B.25, we have for all $i \in [n]$, for all $j \in [n]$,

$$\|z_i - z_j\|_2^2 = f(i-j)$$

for some function f.

We also have

$$\langle z_i, z_j \rangle = 1 - f(i-j)/2 =: g(i-j)$$

as $\|z_i\|_2 = \|z_j\|_2 = 1$.

Then, we have $\forall i, j \in [n]$,

$$\begin{aligned} (QK^\top)_{i,j} &= (ZZ^\top)_{i,j} \\ &= \langle z_i, z_j \rangle \\ &= g(i-j), \end{aligned}$$

where the first two steps from Definition B.29, and the last step from Lemma B.25. We finish the proof by denote b_{i-j} as $g(i-j)$ in Lemma B.27. □

C conv Approximation in Gradient

In Section C.1, we present the basic definitions. In Section C.2, we combine all these definitions to form the loss function. In Section C.3, we analyze the running time. In Section C.4, we present the proof of the main theorem of conv approximation in gradient.

C.1 Definitions

In this section, we let $x, y \in \mathbb{R}^{d^2}$ denote the vectorization of $X, Y \in \mathbb{R}^{d \times d}$. To concisely express the loss function, we define more functions below.

Definition C.1. *Let $u(x)_{j_0} \in \mathbb{R}$ (see Definition 4.3). For each $j_0 \in [n]$, we define $\alpha(x)_{j_0} : \mathbb{R}^{d^2} \to \mathbb{R}$*

$$\alpha(x)_{j_0} := \langle \underbrace{u(x)_{j_0}}_{n \times 1}, \underbrace{\mathbf{1}_n}_{n \times 1} \rangle.$$

Consider $\alpha(x) \in \mathbb{R}^n$ as a vector whose j_0-th entry equals $\alpha(x)_{j_0}$.

Definition C.2. *Let $\alpha(x)_{j_0} \in \mathbb{R}$ (see Definition C.1). Let $u(x)_{j_0} \in \mathbb{R}^n$ (see Definition 4.3). For a fixed $j_0 \in [n]$, we define $f(x)_{j_0} : \mathbb{R}^{d^2} \to \mathbb{R}^n$*

$$f(x)_{j_0} := \underbrace{\alpha(x)_{j_0}^{-1}}_{\text{scalar}} \underbrace{u(x)_{j_0}}_{n \times 1}.$$

Consider $f(x) \in \mathbb{R}^{n \times n}$ as a matrix whose j_0-th row equals $(f(x)_{j_0})^\top$.

Definition C.3. *For a fixed $i_0 \in [d]$, define $h(x)_{i_0} : \mathbb{R}^{d^2} \to \mathbb{R}^n$:*

$$h(y)_{i_0} := \underbrace{A_3}_{n \times d} \underbrace{Y_{*,i_0}}_{d \times 1},$$

where $Y \in \mathbb{R}^{d \times d}$ is the matrix representation of $y \in \mathbb{R}^{d^2}$. Let $h(y) \in \mathbb{R}^{n \times d}$ be a matrix where i_0 column is $h(y)_{i_0}$.

C.2 Loss Functions

Now, we start the construction of the loss function.

Definition C.4. *For each $j_0 \in [n]$, we denote the normalized vector defined by Definition C.2 as $f(x)_{j_0} \in \mathbb{R}^n$. Similarly, for each $i_0 \in [d]$, we define $h(y)_{i_0}$ as specified in Definition C.3.*

Consider every $j_0 \in [n]$, every $i_0 \in [d]$. Let us consider $c(x)_{j_0,i_0} : \mathbb{R}^{d^2} \times \mathbb{R}^{d^2} \to \mathbb{R}$ as follows:

$$c(x)_{j_0,i_0} := \langle f(x)_{j_0}, h(y)_{i_0} \rangle - E_{j_0,i_0}.$$

Here E_{j_0,i_0} is the (j_0,i_0)-th entry of $E \in \mathbb{R}^{n\times d}$ with $j_0 \in [n], i_0 \in [d]$, similar for $\underbrace{c(x)}_{n\times d} = \underbrace{f(x)}_{n\times n}\underbrace{h(y)}_{n\times d} - \underbrace{E}_{n\times d}$.

Definition C.5. *For every $j_0 \in [n]$, for every $i_0 \in [d]$, we define $L(x)_{j_0,i_0}$ to be $:= 0.5c(x)^2_{j_0,i_0}$.*

Definition C.6. *Consider $c(x) \in \mathbb{R}^{n\times d}$ which is described in Definition C.4, and $h(y) \in \mathbb{R}^{n\times d}$ which is defined in Definition C.3. We now define $q(x) \in \mathbb{R}^{n\times n}$*

$$q(x) := \underbrace{c(x)}_{n\times d}\underbrace{h(y)^\top}_{d\times n}$$

Subsequently, we denote the j_0-th row of $q(x) \in \mathbb{R}^{n\times n}$ as $q(x)_{j_0}^\top$.

Definition C.7. *Let $j_0 \in [n]$. We define $p(x)_{j_0} : \mathbb{R}^{d^2} \to \mathbb{R}^n$*

$$\begin{aligned} p(x)_{j_0} :=& (\mathrm{diag}(f(x)_{j_0}) - f(x)_{j_0} f(x)_{j_0}^\top) q(x)_{j_0} \\ =& p_1(x)_{j_0} + p_2(x)_{j_0}, \end{aligned}$$

where

$$\begin{aligned} p_1(x)_{j_0} :=& \mathrm{diag}(f(x)_{j_0}) q(x)_{j_0} \\ p_2(x)_{j_0} :=& f(x)_{j_0} f(x)_{j_0}^\top q(x)_{j_0}. \end{aligned}$$

We establish $p(x) \in \mathbb{R}^{n\times n}$ such that $p(x)_{j_0}^\top$ represents the j_0-th row of $p(x)$. Note that $p_1(x) = f(x) \circ q(x)$.

Lemma C.8. *Let $M \in \mathbb{R}^{n\times n}$ be a casual attention mask defined in Definition 2.2. Let $X \in \mathbb{R}^{n\times n}$, we have*

$$\frac{\mathrm{d}(M \circ X)}{\mathrm{d}X_{i,j}} = M \circ \frac{\mathrm{d}X}{\mathrm{d}X_{i,j}}.$$

Proof. The proof is trivial by element-wise multiplication. □

Lemma C.9 (Gradient computation)**.** *We have $f(x) \in \mathbb{R}^{n\times n}$, $c(x) \in \mathbb{R}^{n\times d}$, $h(y) \in \mathbb{R}^{n\times d}$, $q(x) \in \mathbb{R}^{n\times n}$, and $p(x) \in \mathbb{R}^{n\times n}$ respectively be defined in Definitions C.2, C.4, C.3, C.6, and C.7. Consider $A_1, A_2 \in \mathbb{R}^{n\times d}$ as given and $\mathsf{A} = A_1 \otimes A_2$. We have $L(x)$ be specified in Definition 4.1, and $L(x)_{j_0,i_0}$ is as in Definition C.5.*

Then, we can show that $\frac{\mathrm{d}L(x)}{\mathrm{d}x} = \mathrm{vec}(A_1^\top p(x) A_2)$.

Proof. From the Lemma statement, by Lemma C.8, we have

$$\begin{aligned} & \frac{\mathrm{d}L(x,y)_{j_0,i_0}}{\mathrm{d}x_i} \\ =& c(x,y)_{j_0,i_0} \cdot (\langle M_{j_0,*} \circ f(x)_{j_0} \circ \mathsf{A}_{j_0,i}, h(y)_{i_0} \rangle \\ & - \langle f(x)_{j_0}, h(y)_{i_0} \rangle \cdot \langle M_{j_0,*} \circ f(x)_{j_0}, \mathsf{A}_{j_0,i} \rangle) \\ =& c(x,y)_{j_0,i_0} \cdot (\langle f(x)_{j_0} \circ \mathsf{A}_{j_0,i}, h(y)_{i_0} \rangle \\ & - \langle f(x)_{j_0}, h(y)_{i_0} \rangle \cdot \langle f(x)_{j_0}, \mathsf{A}_{j_0,i} \rangle), \end{aligned} \tag{6}$$

where the first step is from the chain rule and the second step follows from $M_{j_0,*} \circ f(x)_{j_0} = f(x)_{j_0}$.

Note that by Fact B.5, it holds that

$$\begin{aligned} & \langle f(x)_{j_0} \circ \mathsf{A}_{j_0,i}, h(y)_{i_0} \rangle \\ =& \mathsf{A}_{j_0,i}^\top \mathrm{diag}(f(x)_{j_0}) h(y)_{i_0} \end{aligned}$$

and

$$\begin{aligned} & \langle f(x)_{j_0}, v \rangle \cdot \langle f(x)_{j_0}, \mathsf{A}_{j_0,i} \rangle \\ =& \mathsf{A}_{j_0,i}^\top f(x)_{j_0} f(x)_{j_0}^\top h(y)_{i_0} \end{aligned}$$

Therefore, Eq. (6) becomes

$$\begin{aligned} & \frac{\mathrm{d}L(x)_{j_0,i_0}}{\mathrm{d}x_i} \\ =& c(x,y)_{j_0,i_0} \cdot (\mathsf{A}_{j_0,i}^\top \mathrm{diag}(f(x)_{j_0}) h(y)_{i_0} \\ & - \mathsf{A}_{j_0,i}^\top f(x)_{j_0} f(x)_{j_0}^\top h(y)_{i_0}) \\ =& c(x,y)_{j_0,i_0} \cdot \mathsf{A}_{j_0,i}^\top (\mathrm{diag}(f(x)_{j_0}) \\ & - f(x)_{j_0} f(x)_{j_0}^\top) h(y)_{i_0}, \end{aligned} \tag{7}$$

where the last step is by simple algebra.

Let $q(x)_{j_0}$ be defined as in Definition C.6:

$$q(x)_{j_0} := \sum_{i_0=1}^d c(x)_{j_0,i_0} h(y)_{i_0}. \tag{8}$$

Let $p(x)_{j_0}$ be define as in Definition C.7:

$$p(x)_{j_0} := (\mathrm{diag}(f(x)_{j_0}) - f(x)_{j_0} f(x)_{j_0}^\top) q(x)_{j_0}. \tag{9}$$

It holds that

$$\begin{aligned} & \frac{\mathrm{d}L(x)}{\mathrm{d}x} \\ =& \sum_{j_0=1}^n \sum_{i_0=1}^d \frac{\mathrm{d}L(x)_{j_0,i_0}}{\mathrm{d}x} \\ =& \sum_{j_0=1}^n \sum_{i_0=1}^d \underbrace{c(x)_{j_0,i_0}}_{\text{scalar}} \cdot \underbrace{\mathsf{A}_{j_0}^\top}_{d^2\times n} \end{aligned}$$

$$\cdot \underbrace{(\mathrm{diag}(f(x)_{j_0}) - f(x)_{j_0} f(x)_{j_0}^\top)}_{n \times n} \underbrace{h(y)_{i_0}}_{n \times 1}$$
$$= \sum_{j_0=1}^{n} \mathsf{A}_{j_0}^\top (\mathrm{diag}(f(x)_{j_0}) - f(x)_{j_0} f(x)_{j_0}^\top) q(x)_{j_0}$$
$$= \sum_{j_0=1}^{n} \mathsf{A}_{j_0}^\top p(x)_{j_0}$$
$$= \mathrm{vec}(\underbrace{A_1^\top}_{d \times n} \underbrace{p(x)}_{n \times n} \underbrace{A_2}_{n \times d})$$

where the 1st step is because of Definition 4.1, the second step follows from Eq. (7), the third step follows from Eq. (8), the fourth step follows from Eq. (9), and the fifth step follows from Fact F.9. □

C.3 Running Time

In this section, we analyze the running time of the conv approximation approach for computing the training forward pass and backward gradient. We build upon the key definitions and loss functions introduced in the previous sections to derive the running time of the algorithm.

Lemma C.10. *If we have*

- *Define $u(x) \in \mathbb{R}^{n \times n}$ as outlined in Definition 4.3.*
- *Define $f(x) \in \mathbb{R}^{n \times n}$ as specified in Definition C.2.*
- *Define $h(y) \in \mathbb{R}^{n \times d}$ according to Definition C.3.*
- *Suppose $u(x)$ is a k-conv matrix defined in Definition 2.11 with known basis.*

Then, we have

- *For any $w \in \mathbb{R}^n$, we have $f(x) \cdot w \in \mathbb{R}^n$ can be done in $O(kn \log n)$ time.*
- *$h(y)$ can be expiciltiy computed in $\mathcal{T}_{\mathrm{mat}}(n, d, d)$ time.*

Proof. For the first part, by definition of $u(x) \in \mathbb{R}^{n \times n}$, we know that for any vector $w \in \mathbb{R}^n$, we can compute $u(x)w$ in $O(kn \log n)$ time (Claim 2.10). Thus,

$$\begin{aligned} f(x) \cdot w &= \mathrm{diag}(\alpha(x))^{-1} u(x) w \\ &= \mathrm{diag}(u(x) \mathbf{1}_n)^{-1} u(x) w, \end{aligned}$$

which can be done in $O(kn \log n)$ time by Fact B.5.

The second part is trivial by Definition C.3. □

Lemma C.11. *If we have*

- *Define $f(x) \in \mathbb{R}^{n \times n}$ as specified in Definition C.2.*
- *Define $h(y) \in \mathbb{R}^{n \times d}$ according to Definition C.3 and $h(y)$ is known.*
- *Define $c(x) \in \mathbb{R}^{n \times d}$ as outlined in Definition C.4.*
- *Suppose $f(x)w$ takes $O(kn \log n)$ time.*

Then, we can show that

- *$c(x)$ can be expiciltiy computed in $O(knd \log n)$ time.*

Proof. Firstly we can compute $f(x)h(y)$, this can be done in $O(knd \log n)$, since we run $f(x)$ times a vector oracle (Lemma C.10) for d times.

Then do minus $E \in \mathbb{R}^{n \times d}$ matrix. This takes $O(nd)$ time. Thus we complete the proof. □

Lemma C.12. *If the following conditions hold*

- *Let $c(x) \in \mathbb{R}^{n \times d}$ be defined in Definition C.4 and $c(x)$ is known.*
- *Let $h(y) \in \mathbb{R}^{n \times d}$ be defined in Definition C.3 and $h(y)$ is known.*
- *Let $q(x) \in \mathbb{R}^{n \times n}$ be defined in Definition C.6.*

Then, we can show that

- *$q(x)$'s rank-d factorization can be explilcitly computed in $O(nd)$ time.*

Proof. Note that $q(x) = c(x)h(y)^\top$. Since both $c(x)$ and $h(y)$ are known. Thus, the result is trivial. □

Lemma C.13 (Fast computation $p_1(x)$ multiply with a vector)**.** *If the following conditions hold*

- *Let $f(x) \in \mathbb{R}^{n \times n}$ be defined in Definition C.2.*
- *Suppose $f(x)w$ can be done in $O(kn \log n)$ time for any $w \in \mathbb{R}^n$.*
- *Let $q(x)$ denote a rank-τ matrix with known low-rank factorizations.*
- *Let $p_1(x) = f(x) \circ q(x)$.*

Then, we can show

- *For any vector $w \in \mathbb{R}^n$, $p_1(x) \cdot w$ can be computed in $O(\tau kn \log n)$ time*

Proof. Since $q(x) \in \mathbb{R}^{n \times n}$ has rank-τ, we assume that the low-rank factors are $a_1, a_2, \cdots, a_\tau \in \mathbb{R}^n$ and $b_1, b_2, \cdots, b_\tau \in \mathbb{R}^n$. In particular, $q(x)$ can be written as

$$q(x) = \sum_{i=1}^{\tau} a_i b_i^\top$$

Using a standard linear algebra trick, we can show that

$$\begin{aligned} f(x) \circ q(x) &= (f(x)) \circ (\sum_{i=1}^{\tau} a_i b_i^\top) \\ &= \sum_{i=1}^{\tau} (f(x)) \circ (a_i b_i^\top) \\ &= \sum_{i=1}^{\tau} \mathrm{diag}(a_i) f(x) \mathrm{diag}(b_i) \end{aligned}$$

Note that for each $i \in [\tau]$, we can show that $\mathrm{diag}(a_i) f(x) \mathrm{diag}(b_i) w$ can be computed in $O(kn \log n)$ time by Lemma statement. Thus, for any vector $w \in \mathbb{R}^n$, $(f(x) \circ q(x)) \cdot w$ can be computed in $O(\tau kn \log n)$ time. Therefore, we complete the proof.

□

Lemma C.14 (Fast computation for $r(x)$)**.** *If the following conditions hold*

- *Let $r(x)_{j_0} := \langle f(x)_{j_0}, q(x)_{j_0} \rangle$.*
- *Let $f(x) \in \mathbb{R}^{n \times n}$ be defined in Definition C.2.*
- *Suppose $f(x)w$ can be done in $O(kn \log n)$ time for any $w \in \mathbb{R}^n$.*
- *Let $q(x)$ denote a rank-τ matrix with known low-rank factorizations.*

Then, we can show

- *$r(x) \in \mathbb{R}^n$ can be in $O(\tau kn \log n)$ time.*

Proof. Since $q(x) \in \mathbb{R}^{n \times n}$ has rank-τ, we assume that the low-rank factors are $a_1, a_2, \cdots, a_\tau \in \mathbb{R}^n$ and $b_1, b_2, \cdots, b_\tau \in \mathbb{R}^n$, in particular, $q(x)$ can be written as

$$q(x) = \sum_{i=1}^{\tau} a_i b_i^\top$$

Let $q(x) = U_a U_b^\top$. It is easy to see that $f(x)q(x)^\top$ can be written as $f(x)U_b U_a^\top$.

We firstly compute $f(x)U_b$, since U_b has τ columns, each column will take $O(kn \log n)$ time, so in total it takes $O(\tau kn \log n)$ time.

Then, we know that $r(x)_{j_0} = \langle (f(x)U_b)_{j_0,*}, (U_a)_{j_0,*} \rangle$ which takes $O(\tau)$ time per j_0. There are n different j_0, so it takes $O(n\tau)$ time.

Overall it takes $O(\tau kn \log n)$ time.

□

Lemma C.15 (Fat computation for $p_2(x)$)**.** *If the following conditions hold*

- *Assume that $r(x) \in \mathbb{R}^n$ is given.*
- *Let $f(x) \in \mathbb{R}^{n \times n}$ be defined in Definition C.2.*
- *Suppose $f(x)w$ can be done in $O(kn \log n)$ time for any $w \in \mathbb{R}^n$.*
- *Let $p_2(x) = \mathrm{diag}(r(x)) f(x)$ (This is obvious from definition of $r(x)$)*

Then, we can show that

- *For any $w \in \mathbb{R}^n$, $p_2(x) \cdot w$ can be computed $O(kn \log n)$ time.*

Proof. For any vector w, we firstly compute $f(x)w$, then we compute $\mathrm{diag}(r(x))(f(x)w)$.

□

Lemma C.16. *If the following conditions hold*

- *Let $A_1, A_2 \in \mathbb{R}^{n \times d}$ are two given matrices.*
- *Let $p_1(x), p_2(x) \in \mathbb{R}^{n \times n}$ are defined in Definition C.7.*
- *Suppose $p_1(x)w$ takes $\mathcal{T}_{p_1}$ time for any $w \in \mathbb{R}^n$.*
- *Suppose $p_2(x)w$ takes $\mathcal{T}_{p_2}$ time for any $w \in \mathbb{R}^n$.*

Then, we have

- *$\mathrm{vec}(A_1^\top p(x) A_2)$ can be computed in $O(\mathcal{T}_{\mathrm{mat}}(n, d, d) + d(\mathcal{T}_{p_1} + \mathcal{T}_{p_2}))$ time.*

Proof. Firstly, we can compute $p_1(x)A_2$, this takes $d\mathcal{T}_{p_1}$ time.

Second, we can compute $p_2(x)A_2$, this takes $d\mathcal{T}_{p_2}$ time.

Then, we can compute $A_1^\top (p(x)A_2)$, this takes $\mathcal{T}_{\mathrm{mat}}(d, n, d) = O(\mathcal{T}_{\mathrm{mat}}(n, d, d))$.

Putting it all together we complete the proof. □

C.4 Proof of Main Theorem

In this section, we present the formal proof of our main theorem regarding the conv approximation approach for efficiently computing the training forward pass and backward gradient of the attention mechanism.

Theorem C.17. *Suppose $u(x)$ is a k-conv matrix defined in Definition 2.11 with known basis. Then there is an algorithm that runs in time $O(d^2kn\log n)$ time to compute the gradient of attention loss defined in Definition 4.1.*

Proof. We need to choose $\tau = d$, thus total running time is

$$\mathcal{T}_{\mathrm{mat}}(n,d,d) + O(d\tau kn\log n) = O(nd^2k\log n),$$

by putting everything together from Lemma C.9, Lemma C.10, Lemma C.11, Lemma C.12, Lemma C.13, Lemma C.14, Lemma C.15, Lemma C.16. □

Theorem C.18 (Main conv result for training forward and backward gradient (Restatement of Theorem 4.4))**.** *If $u(x)$ is a $1/\operatorname{poly}(n)$-close (T,δ)-non-degenerate k-conv basis matrix as defined in Definition 3.2, where $\delta \geq 0$ and $k, T \in [n]$. Then there are algorithms that run to compute **training forward** in time $O(knd\log n + \mathcal{T}_{\mathrm{mat}}(n,d,d))$ and **backward gradient** in time $O(d^2kn\log n)$ of attention loss (Definition 4.1) approximately up to $1/\operatorname{poly}(n)$ error under ℓ_∞ norm.*

Proof of Theorem 4.4. **Correctness.**

For the forward, we directly get the correctness by Theorem 3.4. For the backward, we directly run error propagation analysis which is similar to (Alman and Song, 2024a) and proof of Lemma B.20.

Running time.

For the forward, by Theorem 3.4, we directly get the running time for $D(X)^{-1}M \circ \exp(A_1XA_2^\top)A_3$ being $O(knd\log n)$. Then, we need $\mathcal{T}_{\mathrm{mat}}(n,d,d)$ time to involve Y and E.

For the backward, by Lemma B.20, we can use Algorithm 2 to get k-conv basis $\widetilde{b}_1,\ldots,\widetilde{b}_k \in \mathbb{R}^n$ and k integers $m_1,m_2,\ldots,m_k$ satisfying $n \geq m_1 > m_2 > \cdots > m_k \geq T$ in time $O(knd\log(n))$. Thus, we finish the proof by Theorem C.17. □

D Low Rank Approximation

We can apply our analysis technique to a low-rank approximation setting in Alman and Song (2023),

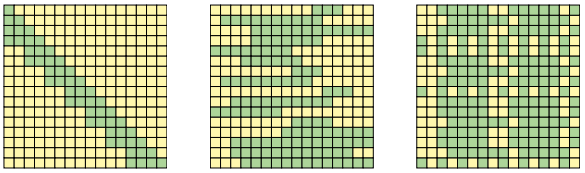

Figure 4: A 16×16 matrix with, left - row change by amortized constant mask (Definition D.1); middle - continuous row mask (Definition D.2); right - distinct 3 rows mask (Definition D.4). Green means 1 and yellow means 0.

which only works on attention approximation without an attention mask. Equipped with our mask analysis trick, we can generalize their results with different kinds of attention masks including the most popular causal attention mask. We first introduce some practical attention masks.

Definition D.1. *Let $B_j \in \mathbb{Z}_{\geq 0}$. We define the row change by amortized constant mask as $W \in \{0,1\}^{n\times n}$, where let $(W^\top)_0 = \mathbf{0}_n$ and $\|(W^\top)_j - (W^\top)_{j-1}\|_1 \leq B_j$ for any $j \in [n]$ and $(W^\top)_j$ is the j-th row of W.*

Definition D.2. *We define the continuous row mask as $W \in \{0,1\}^{n\times n}$, where for each $i \in [n]$, we are given $s_i, t_i \in [n]$ such that $W_{i,j} = 1$ if $s_i \leq j \leq t_i$ and $W_{i,j} = 0$ otherwise.*

Definition D.3. *We define $W \in \{0,1\}^{n\times n}$ as the distinct r columns mask satisfying the following condition. Let $S_1,\cdots,S_r \subseteq [n]$ denote r disjoint subsets and $\cup_{j\in[r]}S_j = [n]$. For any two $i,i' \in S_j$, we have $W_{*,i} = W_{*,i'} \in \mathbb{R}^n$, where $W_{*,i} \in \mathbb{R}^n$ denote the i-th column of $W \in \mathbb{R}^{n\times n}$.*

Definition D.4. *We define $W \in \{0,1\}^{n\times n}$ as the distinct r rows mask satisfying the following condition. Let $S_1,\cdots,S_r \subseteq [n]$ denote r disjoint subsets and $\cup_{j\in[r]}S_j = [n]$. For any two $i,i' \in S_j$, we have $W_{i,*} = W_{i',*} \in \mathbb{R}^n$, where $W_{i,*} \in \mathbb{R}^n$ denotes the i-th row of $W \in \mathbb{R}^{n\times n}$.*

Then, we have the following main results for the low-rank setting. The proof is in Appendix E.2.

Theorem D.5 (Main low-rank result)**.** *Assume the same condition as Lemma E.2. Let $\epsilon \in (0,0.1)$. Let $Q,K,V \in \mathbb{R}^{n\times d}$. Let $U_1,U_2 \in \mathbb{R}^{n\times k}$ be defined in Lemma E.2. Let $W \in \{0,1\}^{n\times n}$ denote a mask matrix. Let $H = \exp(QK^\top/d) \in \mathbb{R}^{n\times n}$, $A = W \circ H \in \mathbb{R}^{n\times n}$ and $D = \operatorname{diag}(A\mathbf{1}_n) \in \mathbb{R}^{n\times n}$. We denote $Y := D^{-1}AV \in \mathbb{R}^{n\times d}$. Let $\widetilde{A} := W \circ U_1U_2^\top$ and $\widetilde{D} := \operatorname{diag}(\widetilde{A}\mathbf{1}_n)$. We denote $\widetilde{Y} := \widetilde{D}^{-1}\widetilde{A}V \in \mathbb{R}^{n\times d}$. Then, we have $\|Y - \widetilde{Y}\|_\infty \leq 4\epsilon\|V\|_\infty$. The time complexity to get $\widetilde{Y}$ is*

- $O(knd)$ *when* W *is a causal mask defined in Definition 2.2.*
- $O(kd\sum_{j=1}^{n} B_j)$ *when* W *is a row change mask defined in Definition D.1.*
- $O(knd\log(n))$ *when* W *is a continuous row mask defined in Definition D.2.*
- $O(rnd)$ *when* W *is a distinct* r *columns / rows mask defined in Definition D.3 / Definition D.4.*

Our Theorem D.5 has the same error guarantee as Alman and Song (2023). For the normal mask, e.g., casual attention mask (Definition 2.2), Theorem D.5 shares the same time complexity as theirs.

E Incorporating Weighted Low Rank Approximation

In Section E.1, we introduce the preliminary for this section. In Section E.2, we present the proof of our main result for the low-rank approximation. In Section E.3, we present the algorithm and its mathematical properties for causal attention mask. In Section E.4, we analyze the algorithm and its mathematical properties for row change by amortized constant mask. In Section E.5, we study the algorithm and its mathematical properties for continuous row mask. In Section E.6, we analyze the property of the mask matrix with r distinct columns or r distinct rows.

E.1 Preliminary

In this section, we introduce the background of the weighted low rank approximation.

Definition E.1 (Definition 3.1 in (Alman and Song, 2023))**.** *Consider a positive integer* $k \geq 1$*. We use* $\epsilon \in (0, 0.1)$ *to represent an accuracy parameter. For* $H \in \mathbb{R}_{\geq 0}^{n\times n}$*, define* $\widetilde{H} \in \mathbb{R}_{\geq 0}^{n\times n}$ *to be an* (ϵ, k)*-approximation of* H *if*

- $\widetilde{H}$ *can be expressed as the product* $U_1 \cdot U_2^\top$ *with some* $U_1, U_2 \in \mathbb{R}^{n\times k}$*, indicating that* $\widetilde{H}$ *has a rank of at most* k*, and*
- $|\widetilde{H}_{i,j} - H_{i,j}| \leq \epsilon \cdot H_{i,j}$ *with any arbitrary* $(i, j) \in [n] \times [n]$*.*

Now, we present a lemma from (Alman and Song, 2023).

Lemma E.2 (Lemma 3.4 in (Alman and Song, 2023))**.** *Let* $Q, K \in \mathbb{R}^{n\times d}$ *satisfy* $\|Q\|_\infty \leq B$ *and* $\|K\|_\infty \leq B$ *respectively for some* $B > 0$ *and* $H \in \mathbb{R}^{n\times n}$ *be defined as* $H := \exp(QK^\top/d)$*. We use* $\epsilon \in (0, 0.1)$ *to represent an accuracy parameter.*

Then, there exist $g > 0$ *with*

$$g = O(\max\{\frac{\log(1/\epsilon)}{\log(\log(1/\epsilon)/B^2)}, B^2\})$$

and $k > 0$ *with*

$$k \leq \binom{2(g+d)}{2g}$$

such that: There exists an (ϵ, k)*-approximation (see Definition E.1) of* $H \in \mathbb{R}^{n\times n}$*, namely* $\widetilde{H} \in \mathbb{R}^{n\times n}$*. Moreover,* U_1 *and* U_2 *defining* $\widetilde{H}$ *is computed in* $O(nk)$ *time.*

In the following lemma, we prove the validity of the statement that if there exists an algorithm whose output is $Y' = (W \circ (U_1U_2^\top))v$ in $O(t)$ time, then there exists an algorithm outputs $Y = D^{-1}(W \circ (U_1U_2^\top))v$ in $O(t + n)$ time. We will combine everything together and show the soundness of this statement later in the proof of Theorem E.4.

Lemma E.3. *Let* $W \in \{0, 1\}^{n\times n}$ *denote any mask matrix. Let* $U_1, U_2 \in \mathbb{R}^{n\times k}$*. Let* $v \in \mathbb{R}^n$*. If there exists an algorithm whose output promises that*

$$Y' = (W \circ (U_1U_2^\top))v,$$

which takes $O(t)$ *time, then, there exists an algorithm promise that*

$$Y = D^{-1}(W \circ (U_1U_2^\top))v$$

where $D := \mathrm{diag}((W \circ (U_1U_2^\top))\mathbf{1}_n) \in \mathbb{R}^{n\times n}$*, which takes* $O(t + n)$ *time.*

Proof. **Correctness.**

Suppose there exists an algorithm whose output is Y' satisfying $Y' = (W \circ (U_1U_2^\top))v$ and takes $O(t)$ time. We denote this algorithm as ALG.

Let $Y' = \textsc{Alg}(U_1, U_2, v)$. Let $\widetilde{Y} = \textsc{Alg}(U_1, U_2, \mathbf{1}_n)$. Then, $Y = \mathrm{diag}(\widetilde{Y})^{-1}Y'$.

Running time.

Computing Y' and $\widetilde{Y}$ takes $O(t)$ time. Computing $Y = \mathrm{diag}(\widetilde{Y})^{-1}Y'$ takes $O(n)$ time. Therefore, it takes $O(t + n)$ time in total. □

E.2 Proof of Main Results

Now, we present our main theorem.

Theorem E.4 (Main low-rank result (Restatement of Theorem D.5))**.** *Assume the same condition as Lemma E.2. Let $\epsilon \in (0, 0.1)$. Let $Q, K, V \in \mathbb{R}^{n \times d}$. Let $U_1, U_2 \in \mathbb{R}^{n \times k}$ be defined in Lemma E.2. Let $W \in \{0,1\}^{n \times n}$ denote a mask matrix. Let $H = \exp(QK^\top/d) \in \mathbb{R}^{n \times n}$, $A = W \circ H \in \mathbb{R}^{n \times n}$ and $D = \mathrm{diag}(A\mathbf{1}_n) \in \mathbb{R}^{n \times n}$. We denote $Y := D^{-1}AV \in \mathbb{R}^{n \times d}$. Let $\widetilde{A} := W \circ U_1 U_2^\top$ and $\widetilde{D} := \mathrm{diag}(\widetilde{A}\mathbf{1}_n)$. We denote $\widetilde{Y} := \widetilde{D}^{-1}\widetilde{A}V \in \mathbb{R}^{n \times d}$. Then, we have*

$$\|Y - \widetilde{Y}\|_\infty \leq 4\epsilon \|V\|_\infty.$$

The time complexity to get $\widetilde{Y}$ is

- *$O(knd)$ when W is a causal mask defined in Definition 2.2.*
- *$O(kd\sum_{j=1}^n B_j)$ when W is a row change mask defined in Definition D.1.*
- *$O(knd\log(n))$ when W is a continuous row mask defined in Definition D.2.*
- *$O(rnd)$ when W is a distinct r columns / rows mask defined in Definition D.3 / Definition D.4.*

Proof of Theorem D.5. **Correctness.**

By Lemma E.2, $U_1 U_2^\top \in \mathbb{R}^{n \times n}$ is an (ϵ, k)-approximation (Definition E.1) of $H \in \mathbb{R}^{n \times n}$. Thus, we have

$$\begin{aligned} |\widetilde{A}_{i,j} - A_{i,j}| &= |(W \circ U_1 U_2^\top)_{i,j} - (W \circ H)_{i,j}| \\ &= W_{i,j} |(U_1 U_2^\top)_{i,j} - H_{i,j}| \\ &\leq W_{i,j} \cdot \epsilon \cdot H_{i,j} \\ &= \epsilon A_{i,j}, \end{aligned}$$

where the first step follows $\widetilde{A} = W \circ U_1 U_2^\top$ and $A = W \circ H$, the second step follows mask is element-wise operation, the third step follows Definition E.1, and the last step follows $A = W \circ H$.

Thus, by Lemma F.6, we get

$$\|Y - \widetilde{Y}\|_\infty \leq 4\epsilon \|V\|_\infty.$$

Running time.

By Lemma E.2, the matrices U_1 and U_2 defining $\widetilde{H}$ can be computed in $O(nk)$ time.

By Lemma E.3, if we can compute $Y' = (W \circ (U_1 U_2^\top))V$ in $O(td)$ time, we can compute $\widetilde{Y}$ in $O(td + nd)$ time.

Finally, we finish the proof by following Lemma E.6 for the causal mask, Lemma E.8 for row change by amortized constant mask, Lemma E.9 for continuous row mask, and Lemma E.12 for distinct r columns mask or distinct r rows mask. □

E.3 Causal Attention Mask

In this section, we present the causal attention mask.

Algorithm 4 Computing $(W \circ (U_1 U_2^\top))v$, where $W \in \{0,1\}^{n \times n}$ is a causal attention mask, as defined in Definition 2.2

procedure CAUSALMASK($U_1 \in \mathbb{R}^{n \times k}, U_2 \in \mathbb{R}^{n \times k}, v \in \mathbb{R}^n$) ▷ Lemma E.6
  $c_0 \leftarrow \mathbf{0}_k$
  for $j = 1 \to n$ **do**
    $b_j \leftarrow \underbrace{(U_2^\top)_j}_{k \times 1} \underbrace{v_j}_{\text{scalar}}$ ▷ Let $(U_2^\top)_j$ denote the j-th row of $U_2 \in \mathbb{R}^{n \times k}$
    $c_j \leftarrow \underbrace{c_{j-1}}_{k \times 1} + \underbrace{b_j}_{k \times 1}$
  end for
  for $j = 1 \to n$ **do**
    $Y_j \leftarrow \langle \underbrace{(U_1^\top)_j}_{k \times 1}, \underbrace{c_j}_{k \times 1} \rangle$
  end for
  return Y ▷ $Y \in \mathbb{R}^n$
end procedure

Lemma E.5. *Let $W \in \{0,1\}^{n \times n}$ be a mask. Let S_j denote the support set of each row of W, for each $j \in [n]$, i.e., $S_j = \{k | W_{j,k} = 1\}$. Let $U_1, U_2 \in \mathbb{R}^{n \times k}$. Let $v \in \mathbb{R}^n$. Let $Y = (W \circ (U_1 U_2^\top))v$. Then, we have*

$$Y_j = \langle (U_1^\top)_j, \sum_{l \in S_j} (U_2^\top)_l v_l \rangle.$$

Proof. By simple algebra, we have

$$\begin{aligned} Y_j &= ((W \circ (U_1 U_2^\top))v)_j \\ &= \langle (U_1^\top)_j, \sum_{l \in S_j} (U_2^\top)_l v_l \rangle. \end{aligned}$$

□

Lemma E.6. *Let $W \in \{0,1\}^{n \times n}$ be a causal attention mask defined in Definition 2.2. Let $U_1, U_2 \in \mathbb{R}^{n \times k}$. Let $v \in \mathbb{R}^n$. Then, there exists an algorithm (see Algorithm 4) whose output*

promises that

$$Y = (W \circ (U_1 U_2^\top))v,$$

which takes $O(nk)$ time.

Proof. Let $(U_2^\top)_j$ denote the j-th row of U_2.

Correctness.

Let S_j be the support set defined in Lemma E.5. Note that for the causal attention mask, we have $S_j = [j]$ for any $j \in [n]$. Thus, by Lemma E.5, we have

$$\begin{aligned} Y_j &= \langle (U_1^\top)_j, \sum_{l \in [j]} (U_2^\top)_l v_l \rangle \\ &= \langle (U_1^\top)_j, c_j \rangle. \end{aligned}$$

Running time.

Computing $(U_2^\top)_j v_j$, for all $j \in [n]$ takes $O(nk)$ time.

Note that by the definition of inner product

$$\langle (U_1^\top)_j, c_j \rangle = (U_1^\top)_j^\top c_j.$$

Therefore, it also takes $O(nk)$ to compute $(U_1^\top)_j^\top c_j$ for all $j \in [n]$.

Therefore, it takes $O(nk)$ times in total. □

E.4 Row Change by Amortized Constant Mask

In this section, we analyze the row change by amortized constant mask.

Claim E.7. *Let $W \in \{0,1\}^{n \times n}$ be the causal attention mask defined in Definition 2.2. Then we have W is a row change by amortized constant mask defined in Definition D.1, where $B_j = 1$, $\forall j \in [n]$.*

Proof. The proof directly follows the two Definitions. □

Lemma E.8. *Let $B \in \mathbb{Z}_{\geq 0}$ and let $W \in \{0,1\}^{n \times n}$ be a row change by amortized constant mask defined in Definition D.1. Let $S_0 = \emptyset$. Let S_j be the support set of each row of W, for each $j \in [n]$, i.e., $S_j = \{k | W_{j,k} = 1\}$. We define $B_j := |(S_j \backslash S_{j-1}) \cup (S_{j-1} \backslash S_j)|$. Let $U_1, U_2 \in \mathbb{R}^{n \times k}$. Let $v \in \mathbb{R}^n$. Then, there exists an algorithm (see Algorithm 5) whose output promises that*

$$Y = (W \circ (U_1 U_2^\top))v,$$

which takes $O(k \sum_{j=1}^n B_j)$ time.

Algorithm 5 Computing $(W \circ (U_1 U_2^\top))v$, where $W \in \{0,1\}^{n \times n}$ is a row change by amortized constant mask, as defined in Definition D.1

procedure ConstantMask($U_1 \in \mathbb{R}^{n \times k}, U_2 \in \mathbb{R}^{n \times k}, v \in \mathbb{R}^n$) ▷ Lemma E.8
  $c_0 \leftarrow \mathbf{0}_k$, $S_0 \leftarrow \emptyset$
  for $j = 1 \to n$ **do**
    Precompute indices set $Q_j^+ \leftarrow S_j \backslash S_{j-1}$ ▷ Let S_j denote the support set of the j-th row
    Precompute indices set $Q_j^- \leftarrow S_{j-1} \backslash S_j$
    $c_j \leftarrow c_{j-1}$
    for $i \in Q_j^+ \cup Q_j^-$ **do** ▷ $|Q_j^+ \cup Q_j^-| = B_j$
      $b_i \leftarrow \underbrace{(U_2^\top)_i}_{k \times 1} \underbrace{v_i}_{\text{scalar}}$ ▷ Let $(U_2^\top)_i$ denote the i-th row of $U_2 \in \mathbb{R}^{n \times k}$
      if $i \in Q_j^+$ **then**
        $c_j \leftarrow c_j + b_i$
      else if $i \in Q_j^-$ **then**
        $c_j \leftarrow c_j - b_i$
      end if
    end for
  end for
  for $j = 1 \to n$ **do**
    $Y_j \leftarrow \langle \underbrace{(U_1^\top)_j}_{k \times 1}, \underbrace{c_j}_{k \times 1} \rangle$
  end for
  return Y ▷ $Y \in \mathbb{R}^n$
end procedure

Proof. **Correctness.**

By Lemma E.5, we have

$$Y_j = \langle (U_1^\top)_j, \sum_{l \in S_j} (U_2^\top)_l v_l \rangle.$$

We will prove it by induction. It is obvious that base case Y_1 is correct, because $S_0 = \emptyset$.

For a fixed j, we suppose Y_j has the correct answer. This means c_j is correct for that j, i.e., $c_j = \sum_{l \in S_j} b_l = \sum_{l \in S_j} (U_2^\top)_l v_l$.

Now we use Q_{j+1}^+ and Q_{j+1}^- to generate c_{j+1} by adding terms in Q_{j+1}^+ and deleting terms in Q_{j+1}^-,

$$\begin{aligned} & c_{j+1} \\ = & \sum_{l \in S_j} b_l - \sum_{l \in S_j \backslash S_{j+1}} b_l + \sum_{l \in S_{j+1} \backslash S_j} b_l \\ = & \sum_{l \in S_j \cap S_{j+1}} b_l + \sum_{l \in S_j \backslash S_{j+1}} b_l \end{aligned}$$

$$
\begin{aligned}
& - \sum_{l \in S_j \setminus S_{j+1}} b_l + \sum_{l \in S_{j+1} \setminus S_j} b_l \\
= & \sum_{l \in S_j \cap S_{j+1}} b_l + \sum_{l \in S_{j+1} \setminus S_j} b_l \\
= & \sum_{l \in S_{j+1}} b_l,
\end{aligned}
$$

where the first step follows Algorithm 5 line 10 and line 12, the second step follows $S_j = (S_j \cap S_{j+1}) \cup (S_j \setminus S_{j+1})$, $(S_j \cap S_{j+1})$ and $(S_j \setminus S_{j+1})$ are disjoint, the third step follows simple algebra, and the last step follows the as the second step.

Therefore, we have c_{j+1} is correct, i.e., $c_{j+1} = \sum_{l \in S_{j+1}} b_l = \sum_{l \in S_{j+1}} (U_2^\top)_l v_l$. Thus, Y_{j+1} is also correct by Lemma E.5. Finally, we finish proving the correctness by math induction.

Running time.

Note that there are two for-loops in this algorithm. Inside the inner for-loops, it takes $O(k)$ time to compute

$$
b_i = \underbrace{(U_2^\top)_i}_{k \times 1} \underbrace{v_i}_{\text{scalar}}.
$$

The inner for-loop has $|Q_j^+ \cup Q_j^-| = B_j$ iterations, and the outer for-loop has n iterations.

Therefore, it takes $O(k \sum_{j=1}^n B_j)$ time in total. □

E.5 Continuous Row Mask

In this section, we study the continuous row mask.

Lemma E.9. *Let $W \in \{0,1\}^{n \times n}$ denote a continuous row mask defined in Definition D.2. Let $U_1, U_2 \in \mathbb{R}^{n \times k}$. Let $v \in \mathbb{R}^n$. Then, there exists an algorithm (see Algorithm 6) whose output promises that*

$$
Y = (W \circ (U_1 U_2^\top))v,
$$

which takes $O(nk \log n)$ time.

Proof. The correctness is trivially from the construction of the segment tree.

The running time is dominated by $O(nk \log n)$. This time comes from two parts, where the first is from building the segment tree by $O(nk)$, and the second part is from for-loop by $O(nk \log n)$. □

E.6 Distinct r Columns or Rows

Now, we analyze the mask matrix with r distinct columns.

Algorithm 6 Computing $(W \circ (U_1 U_2^\top))v$, where $W \in \{0,1\}^{n \times n}$ is a continuous row mask, as defined in Definition D.2

```
1: procedure ContinuousMask(U_1 ∈ R^{n×k}, U_2 ∈ R^{n×k}, v ∈ R^n)   ▷ Lemma E.9
2:     c_0 ← 0_k
3:     Build segment tree T based on {(U_2^⊤)_i v_i}_{i∈[n]}
4:     for j = 1 → n do
5:         Get at most O(log n) vectors from T (each one is a continuous summation of 2^t entries)
6:         Compute c_j based on the above vectors
7:     end for
8:     for j = 1 → n do
9:         Y_j ← ⟨ underbrace((U_1^⊤)_j, k×1), underbrace(c_j, k×1) ⟩
10:    end for
11: return Y                                                          ▷ Y ∈ R^n
12: end procedure
```

Lemma E.10. *Let W be the distinct r columns mask defined in Definition D.3. Let $S_1, \cdots, S_r \subseteq [n]$ denote r disjoint subsets and $\cup_{j \in [r]} S_j = [n]$ be defined in Definition D.3. Let $h : [r] \to [n]$ denote that $h(j) \in S_j$ and $h(j)$ is the smallest index in S_j.*

Then we can show

$$
\begin{aligned}
& (\underbrace{W}_{n \times n} \circ (\underbrace{U_1}_{n \times k} \underbrace{U_2^\top}_{k \times n})) \underbrace{v}_{n \times 1} \\
= & \sum_{j=1}^r \underbrace{\mathrm{diag}(W_{*,h(j)})}_{n \times n} \underbrace{U_1}_{n \times k} \underbrace{(U_2^\top)_{*,S_j}}_{k \times |S_j|} \underbrace{v_{S_j}}_{|S_j| \times 1}
\end{aligned}
$$

Proof. We can show that

$$
\begin{aligned}
\mathrm{LHS} = & \sum_{i=1}^n (W \circ (U_1 U_2^\top))_{*,i} \cdot v_i \\
= & \sum_{i=1}^n (W_{*,i} \circ (U_1 U_2^\top)_{*,i}) v_i \\
= & \sum_{i=1}^n \mathrm{diag}(W_{*,i}) (U_1 U_2^\top)_{*,i} v_i \\
= & \sum_{i=1}^n \mathrm{diag}(W_{*,i}) U_1 (U_2^\top)_{*,i} v_i \\
= & \sum_{j=1}^r \mathrm{diag}(W_{*,h(j)}) U_1 (U_2^\top)_{*,S_j} v_{S_j},
\end{aligned}
$$

where the first step follows from the left hand side of the equation in the lemma statement, the second step follows from the definition of the Hadamard product, the third step follows from Fact B.5, the fourth step follows from simple algebra, and the last step follows from the fact that for any two $i, i' \in S_j$, we have $W_{*,i} = W_{*,i'} \in \mathbb{R}^n$ (see from the lemma statement). □

Now, we analyze the mask matrix with r distinct rows.

Lemma E.11. *Let W be the distinct r rows mask defined in Definition D.4. Let $S_1, \cdots, S_r \subseteq [n]$ denote r disjoint subsets and $\cup_{j \in [r]} S_j = [n]$ be defined in Definition D.4. Let $h : [r] \to [n]$ denote that $h(j) \in S_j$ and $h(j)$ is the smallest index in S_j.*

Then, we can show that

$$\begin{aligned} &(\underbrace{W}_{n \times n} \circ (\underbrace{U_1}_{n \times k} \underbrace{U_2^\top}_{k \times n})) \underbrace{v}_{n \times 1} \\ = &\sum_{j=1}^r \underbrace{\mathrm{diag}(e_{S_j})}_{n \times n} \underbrace{U_1}_{n \times k} \underbrace{U_2^\top}_{k \times n} \underbrace{\mathrm{diag}(W_{h(j),*})}_{n \times n} \underbrace{v}_{n \times 1} \end{aligned}$$

Proof. It suffices to show

$$\begin{aligned} &(\underbrace{W}_{n \times n} \circ (\underbrace{U_1}_{n \times k} \underbrace{U_2^\top}_{k \times n})) \\ = &\sum_{j=1}^r \underbrace{\mathrm{diag}(e_{S_j})}_{n \times n} \underbrace{U_1}_{n \times k} \underbrace{U_2^\top}_{k \times n} \underbrace{\mathrm{diag}(W_{h(j),*})}_{n \times n}. \end{aligned} \tag{10}$$

We have

$$\begin{aligned} & (W \circ (U_1 U_2^\top)) \\ = & ((U_1 U_2^\top) \circ W) \\ = & \sum_{i=1}^n (\mathrm{diag}(e_i)(U_1 U_2^\top) \circ W)_{i,*} \\ = & \sum_{i=1}^n (\mathrm{diag}(e_i)(U_1 U_2^\top) \circ W_{i,*}) \\ = & \sum_{i=1}^n (\mathrm{diag}(e_i)(U_1 U_2^\top) \, \mathrm{diag}(W_{i,*})) \\ = & \sum_{j=1}^n \mathrm{diag}(e_{S_j}) U_1 U_2^\top \, \mathrm{diag}(W_{h(j),*}), \end{aligned}$$

where the first step follows from the definition of the Hadamard product, the second step follows from the property of $\mathrm{diag}(e_i)$ that for any matrix A, $\mathrm{diag}(e_i)A$ preserves the i-th row of A and set other rows to 0, the third step follows from simple algebra, the fourth step follows from Fact B.5, and the last step follows from the lemma statement that for any two $i, i' \in S_j$, we have $W_{i,*} = W_{i',*} \in \mathbb{R}^n$.

Therefore, we have shown Eq. (10), which completes the proof. □

Lemma E.12. *Let $W \in \{0,1\}^{n \times n}$ be a distinct r columns mask defined in Definition D.3 or a distinct r rows mask defined in Definition D.4. Let $U_1, U_2 \in \mathbb{R}^{n \times k}$. Let $v \in \mathbb{R}^n$. Then, there exists an algorithm whose output promises that*

$$Y = (W \circ (U_1 U_2^\top))v,$$

which takes $O(nkr)$ time.

Proof. The correctness and running time is directly follows Lemma E.10 for the column case and Lemma E.11 for the row case. □

F Supporting Lemmas and Technical Results

In Section F.1, we present the matrix and vector properties. In Section F.2, we analyze and develop the tools for error analysis. In Section F.3, we provide some tools for tensor calculation.

F.1 Matrix and Vector Properties

Lemma F.1 (Restatement of Lemma 2.12)**.** *For any lower triangular matrix $H \neq \mathbf{0}_{n \times n} \in \mathbb{R}^{n \times n}$, there exists a unique $k \in [n]$ such that H is a matrix with k-conv basis.*

Proof of Lemma 2.12. It suffices to show that any arbitrary $H \in \mathbb{R}^{n \times n} \setminus \{\mathbf{0}_{n \times n}\}$ has at least 1 conv basis and at most n conv basis.

As $H \neq \mathbf{0}_{n \times n}$, it must have at least 1 conv basis, and we proved the first part.

Now, we prove the second part by math induction.

Let $i \in \{0, \ldots, n-1\}$. For any lower triangular matrix $G \in \mathbb{R}^{n \times n}$, we have

$$G = \begin{bmatrix} \mathbf{0}_{i \times i} & \mathbf{0}_{i \times (n-i)} \\ \mathbf{0}_{(n-i) \times i} & G_{(i+1):n,(i+1):n} \end{bmatrix}.$$

Let G_{i+1} be the $i+1$-th column of $G \in \mathbb{R}^{n \times n}$. Let $\widetilde{G}_{i+1} \in \mathbb{R}^n$ satisfy, for any $j \in [n]$, $(\widetilde{G}_{i+1})_j = (G_{i+1})_{i+j}$ when $i + j \leq n$ and $(\widetilde{G}_{i+1})_j = (G_{i+1})_{i+j-n}$ otherwise. Then, there exists lower triangular matrix $G' \in \mathbb{R}^{(n-i-1) \times (n-i-1)}$ such that

$$G - \mathsf{conv}(\widetilde{G}_{i+1}, n-i)$$

$$
\begin{aligned}
&= \begin{bmatrix} \mathbf{0}_{i\times i} & \mathbf{0}_{i\times 1} & \mathbf{0}_{i\times(n-i-1)} \\ \mathbf{0}_{1\times i} & G_{i+1,i+1} & \mathbf{0}_{1\times(n-i-1)} \\ \mathbf{0}_{(n-i-1)\times i} & G_{(i+2):n,(i+1)} & G_{(i+2):n,(i+2):n} \end{bmatrix} \\
&- \begin{bmatrix} \mathbf{0}_{i\times i} & \mathbf{0}_{i\times 1} & \mathbf{0}_{i\times(n-i-1)} \\ \mathbf{0}_{1\times i} & G_{i+1,i+1} & \mathbf{0}_{1\times(n-i-1)} \\ \mathbf{0}_{(n-i-1)\times i} & G_{(i+2):n,(i+1)} & G' \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{0}_{i\times i} & \mathbf{0}_{i\times 1} & \mathbf{0}_{i\times(n-i-1)} \\ \mathbf{0}_{1\times i} & \mathbf{0}_{1\times 1} & \mathbf{0}_{1\times(n-i-1)} \\ \mathbf{0}_{(n-i-1)\times i} & \mathbf{0}_{(n-i-1)\times 1} & G_{(i+2):n,(i+2):n} - G' \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{0}_{(i+1)\times(i+1)} & \mathbf{0}_{(i+1)\times(n-i-1)} \\ \mathbf{0}_{(n-i-1)\times(i+1)} & G_{(i+2):n,(i+2):n} - G' \end{bmatrix},
\end{aligned}
$$

where the first step follows from the fact that G is a lower triangular matrix and Definition 2.9, the second step follows from simple algebra, and the last step follows from simple algebra.

As G and G' are lower triangular matrices, we have that $G - \mathsf{conv}(\widetilde{G}_{i+1}, n-i)$ is a lower triangular matrix. Thus, we proved the following statement.

For any lower triangular matrix $G \in \mathbb{R}^{n\times n}$ whose first i columns all are zeros, there exists a basis $\mathsf{conv}(b,m)$ such that $G - \mathsf{conv}(b,m) \in \mathbb{R}^{n\times n}$ is a lower triangular matrix whose first $i+1$ columns all are zeros.

As $H \in \mathbb{R}^{n\times n}$ is a lower triangular matrix whose first 0 columns all are zeros, we finish the proof by math induction, i.e., repeat the above process at most n times. □

Lemma F.2. *For any matrix $G \in \mathbb{R}^{n\times n}$ and vector $v \in \mathbb{R}^n$, we have*

$$\|Gv\|_1 \leq \|G\|_1 \cdot \|v\|_\infty.$$

Proof. We have

$$
\begin{aligned}
\|Gv\|_1 &= \sum_{i\in[n]} |\sum_{j\in[n]} G_{i,j} v_j| \\
&\leq \sum_{i\in[n]} \sum_{j\in[n]} |G_{i,j} v_j| \\
&\leq \sum_{i\in[n]} \sum_{j\in[n]} |G_{i,j}| \|v\|_\infty \\
&= \|G\|_1 \cdot \|v\|_\infty,
\end{aligned}
$$

where the first step follows the Definition of vector ℓ_1 norm, the second steps follow $|a+b| \leq |a| + |b|$, the third steps follow simple algebra, and the last step follow the Definition of matrix ℓ_1 norm. □

F.2 Tools for Error Analysis

Lemma F.3. *Let $\epsilon \geq 0$. Let $x_1, x_2 \in \mathbb{R}$. We have*

$$
\begin{aligned}
&|\exp(x_1) - \exp(x_2)| \\
\leq\ &\exp(\min\{x_1, x_2\})(\exp(|x_1 - x_2|) - 1).
\end{aligned}
$$

Proof. It is trivial by $\exp(a+b) = \exp(a)\exp(b)$. □

Lemma F.4. *Let $V \in \mathbb{R}^{n\times d}$. Let $H, \widetilde{H} \in \mathbb{R}^{n\times n}$, and satisfy $\|H - \widetilde{H}\|_\infty \leq \epsilon$, where $\epsilon \geq 0$. Let $A = \exp(H)$, $\widetilde{A} = \exp(\widetilde{H})$ and $D = \mathrm{diag}(A\mathbf{1}_n)$, $\widetilde{D} = \mathrm{diag}(\widetilde{A}\mathbf{1}_n)$. Then, we have*

$$\|D^{-1}AV - \widetilde{D}^{-1}\widetilde{A}V\|_\infty \leq 2(\exp(\epsilon) - 1)\|V\|_\infty.$$

Proof. By triangle inequality, we have $\|D^{-1}AV - \widetilde{D}^{-1}\widetilde{A}V\|_\infty = \|D^{-1}AV - \widetilde{D}^{-1}AV\|_\infty + \|\widetilde{D}^{-1}AV - \widetilde{D}^{-1}\widetilde{A}V\|_\infty$, where the first step follows simple algebra, and the last step follows triangle inequality.

For the first part, for any $i \in [n], j \in [n]$, we have

$$
\begin{aligned}
&|(D^{-1}AV - \widetilde{D}^{-1}AV)_{i,j}| \\
=\ &|\sum_{l=1}^n (D_{i,i}^{-1} - \widetilde{D}_{i,i}^{-1}) A_{i,l} V_{l,j}| \\
\leq\ &\sum_{l=1}^n |(D_{i,i}^{-1} - \widetilde{D}_{i,i}^{-1}) A_{i,l}| \cdot \|V\|_\infty \\
=\ &\sum_{l=1}^n |\frac{D_{i,i} - \widetilde{D}_{i,i}}{D_{i,i}\widetilde{D}_{i,i}}| \cdot A_{i,l} \cdot \|V\|_\infty \\
=\ &\sum_{l=1}^n |\sum_{k=1}^n \exp(H_{i,k}) - \sum_{k=1}^n \exp(\widetilde{H}_{i,k})| \\
&\cdot \frac{A_{i,l}}{D_{i,i}\widetilde{D}_{i,i}} \cdot \|V\|_\infty \\
\leq\ &\sum_{l=1}^n \sum_{k=1}^n |\exp(H_{i,k}) - \exp(\widetilde{H}_{i,k})| \\
&\cdot \frac{A_{i,l}}{D_{i,i}\widetilde{D}_{i,i}} \cdot \|V\|_\infty \\
\leq\ &(\exp(\epsilon) - 1) \sum_{l=1}^n \sum_{k=1}^n \exp(\widetilde{H}_{i,k}) \\
&\cdot \frac{A_{i,l}}{D_{i,i}\widetilde{D}_{i,i}} \cdot \|V\|_\infty \\
=\ &(\exp(\epsilon) - 1)\|V\|_\infty,
\end{aligned}
$$

where the first step follows simple algebra, the second step follows triangle inequality, the third step follows simple algebra, the fourth step follows

$D = \mathrm{diag}(A\mathbf{1}_n)$, $\widetilde{D} = \mathrm{diag}(\widetilde{A}\mathbf{1}_n)$, $A = \exp(H)$, $\widetilde{A} = \exp(\widetilde{H})$, the fifth steps follows triangle inequality, the sixth step follows Lemma F.3 and the last step follows $\widetilde{D}_{i,i} = \sum_{k=1}^n \exp(\widetilde{H}_{i,k})$ and $D_{i,i} = \sum_{l=1}^n A_{i,l}$.

For the second part, for any $i \in [n], j \in [n]$, we have

$$\begin{aligned}
&|(\widetilde{D}^{-1}AV - \widetilde{D}^{-1}\widetilde{A}V)_{i,j}| \\
=& |\sum_{l=1}^n \widetilde{D}_{i,i}^{-1}(A_{i,l} - \widetilde{A}_{i,l})V_{l,j}| \\
\leq& \sum_{l=1}^n \widetilde{D}_{i,i}^{-1}|A_{i,l} - \widetilde{A}_{i,l}| \cdot \|V\|_\infty \\
=& \sum_{l=1}^n \widetilde{D}_{i,i}^{-1}|\exp(H_{i,l}) - \exp(\widetilde{H}_{i,l})| \cdot \|V\|_\infty \\
\leq& (\exp(\epsilon) - 1)\sum_{l=1}^n \widetilde{D}_{i,i}^{-1}\exp(\widetilde{H}_{i,l}) \cdot \|V\|_\infty \\
=& (\exp(\epsilon) - 1)\|V\|_\infty,
\end{aligned}$$

where the first step follows simple algebra, the second step follows triangle inequality, the third step follows $A = \exp(H)$, $\widetilde{A} = \exp(\widetilde{H})$, the fourth step follows Lemma F.3, and the last step follows $\widetilde{D}_{i,i} = \sum_{l=1}^n \exp(\widetilde{H}_{i,l})$.

Thus, we combine two terms,

$$\|D^{-1}AV - \widetilde{D}^{-1}\widetilde{A}V\|_\infty \leq 2(\exp(\epsilon) - 1)\|V\|_\infty.$$

□

Lemma F.5. *Let $a, b \geq 0$ and $\epsilon \in (0, 0.1)$. If $|a - b| \leq \epsilon a$, then $|a - b| \leq 2\epsilon \min\{a, b\}$.*

Proof. It is trivial by considering two cases when $b \geq a$ and $b < a$. □

Lemma F.6. *Let $A, \widetilde{A} \in \mathbb{R}_{\geq 0}^{n \times n}$, and satisfy $|\widetilde{A}_{i,j} - A_{i,j}| \leq \epsilon \cdot A_{i,j}$ for all $(i, j) \in [n]^2$, where $\epsilon \in (0, 0.1)$. Let $D = \mathrm{diag}(A\mathbf{1}_n)$ and $\widetilde{D} = \mathrm{diag}(\widetilde{A}\mathbf{1}_n)$. Then, we have*

$$\|D^{-1}AV - \widetilde{D}^{-1}\widetilde{A}V\|_\infty \leq 4\epsilon\|V\|_\infty.$$

Proof. By triangle inequality, we have

$$\begin{aligned}
&\|D^{-1}AV - \widetilde{D}^{-1}\widetilde{A}V\|_\infty \\
\leq& \|D^{-1}AV - \widetilde{D}^{-1}AV\|_\infty + \|\widetilde{D}^{-1}AV - \widetilde{D}^{-1}\widetilde{A}V\|_\infty,
\end{aligned}$$

where the first step follows simple algebra, and the last step follows triangle inequality.

For the first part, for any $i \in [n], j \in [n]$, we have

$$\begin{aligned}
&|(D^{-1}AV - \widetilde{D}^{-1}AV)_{i,j}| \\
=& |\sum_{l=1}^n (D_{i,i}^{-1} - \widetilde{D}_{i,i}^{-1})A_{i,l}V_{l,j}| \\
\leq& \sum_{l=1}^n |(D_{i,i}^{-1} - \widetilde{D}_{i,i}^{-1})A_{i,l}| \cdot \|V\|_\infty \\
=& \sum_{l=1}^n |\frac{D_{i,i} - \widetilde{D}_{i,i}}{D_{i,i}\widetilde{D}_{i,i}}| \cdot A_{i,l} \cdot \|V\|_\infty \\
=& \sum_{l=1}^n |\sum_{k=1}^n A_{i,k} - \sum_{k=1}^n \widetilde{A}_{i,k}| \cdot \frac{A_{i,l}}{D_{i,i}\widetilde{D}_{i,i}} \cdot \|V\|_\infty \\
\leq& \sum_{l=1}^n \sum_{k=1}^n |A_{i,k} - \widetilde{A}_{i,k}| \cdot \frac{A_{i,l}}{D_{i,i}\widetilde{D}_{i,i}} \cdot \|V\|_\infty \\
\leq& 2\epsilon \sum_{l=1}^n \sum_{k=1}^n \widetilde{A}_{i,k} \cdot \frac{A_{i,l}}{D_{i,i}\widetilde{D}_{i,i}} \cdot \|V\|_\infty \\
=& 2\epsilon\|V\|_\infty,
\end{aligned}$$

where the first step follows simple algebra, the second step follows triangle inequality, the third step follows simple algebra, the fourth step follows $D = \mathrm{diag}(A\mathbf{1}_n)$, $\widetilde{D} = \mathrm{diag}(\widetilde{A}\mathbf{1}_n)$, the fifth step follows triangle inequality, the sixth step follows Lemma F.5 and the last step follows $\widetilde{D}_{i,i} = \sum_{k=1}^n \widetilde{A}_{i,k}$ and $D_{i,i} = \sum_{l=1}^n A_{i,l}$.

For the second part, for any $i \in [n], j \in [n]$, we have

$$\begin{aligned}
&|(\widetilde{D}^{-1}AV - \widetilde{D}^{-1}\widetilde{A}V)_{i,j}| \\
=& |\sum_{l=1}^n \widetilde{D}_{i,i}^{-1}(A_{i,l} - \widetilde{A}_{i,l})V_{l,j}| \\
\leq& \sum_{l=1}^n \widetilde{D}_{i,i}^{-1}|A_{i,l} - \widetilde{A}_{i,l}| \cdot \|V\|_\infty \\
\leq& 2\epsilon \sum_{l=1}^n \widetilde{D}_{i,i}^{-1}\widetilde{A}_{i,l} \cdot \|V\|_\infty \\
=& 2\epsilon\|V\|_\infty,
\end{aligned}$$

where the first step follows simple algebra, the second step follows triangle inequality, the third step follows Lemma F.5, and the last step follows $\widetilde{D}_{i,i} = \sum_{l=1}^n \widetilde{A}_{i,l}$.

Thus, we combine two terms,

$$\|D^{-1}AV - \widetilde{D}^{-1}\widetilde{A}V\|_\infty \leq 4\epsilon\|V\|_\infty.$$

□

F.3 Tensor Tools for Gradient Computation

Fact F.7 (Fact A.3 on page 15 of (Li et al., 2024c), also see (Bürgisser et al., 2013; Bläser, 2013) for more detail)**.** *We can show that*

$$\begin{aligned}\mathcal{T}_{\mathrm{mat}}(a,b,c) &= O(\mathcal{T}_{\mathrm{mat}}(a,c,b)) \\ &= O(\mathcal{T}_{\mathrm{mat}}(b,a,c)) \\ &= O(\mathcal{T}_{\mathrm{mat}}(b,c,a)) \\ &= O(\mathcal{T}_{\mathrm{mat}}(c,a,b)) \\ &= O(\mathcal{T}_{\mathrm{mat}}(c,b,a)).\end{aligned}$$

Fact F.8. *Let $a \in \mathbb{R}^n, b \in \mathbb{R}^d$. We have*

$$\mathrm{vec}(ab^\top) = a \otimes b$$

Proof. We can show

$$\begin{aligned}\mathrm{vec}(ab^\top) = &\ \mathrm{vec}(\begin{bmatrix} a_1 b^\top \\ a_2 b^\top \\ \dots \\ a_n b^\top \end{bmatrix}) \\ = &\ [a_1 b^\top, a_2 b^\top, \dots, a_n b^\top]^\top \\ = &\ a \otimes b\end{aligned}$$

where the first step follows from the definition of outer product, the second step follows from the definition of vectorization operator $\mathrm{vec}(\cdot)$ which stacks rows of a matrix into a column vector, and the last step follows from the definition of the Kronecker product. □

Fact F.9 (Tensor-trick on page 3 of (Gao et al., 2023a), also see (Diao et al., 2018) for more detail)**.** *Given matrices $A_1 \in \mathbb{R}^{n_1 \times d_1}, A_2 \in \mathbb{R}^{n_2 \times d_2}$ and $X \in \mathbb{R}^{d_1 \times d_2}$, the well-known tensor-trick suggests that $\mathrm{vec}(A_1 X A_2^\top) = (A_1 \otimes A_2)\,\mathrm{vec}(X) \in \mathbb{R}^{n_1 n_2}$.*

Proof. We can show

$$\begin{aligned}\mathrm{vec}(A_1 X A_2^\top) = &\ \sum_{i=1}^{d_1}\sum_{j=1}^{d_2} X_{i,j}\,\mathrm{vec}(A_{1,*,i}(A_{2,*,j})^\top) \\ = &\ \sum_{i=1}^{d_1}\sum_{j=1}^{d_2} X_{i,j}(\underbrace{A_{1,*,i}}_{n_1\times 1} \otimes \underbrace{A_{2,*,j}}_{n_2\times 1}) \\ = &\ \sum_{i=1}^{d_1}(\underbrace{A_{1,*,i}}_{n_1\times 1} \otimes \underbrace{A_2}_{n_2\times d_2})\,\underbrace{X_{i,*}}_{d_2\times 1} \\ = &\ (A_1 \otimes A_2)\,\mathrm{vec}(X)\end{aligned}$$

where the first step follows from that matrix can be written as a summation of vectors, the second step follows from Fact F.8, the third step follows from that matrix can be written as a summation of vectors, and the last step follows from the definition of vectorization operator $\mathrm{vec}(\cdot)$. □

G Potential Risks

This work introduces a theoretical and algorithmic framework for accelerating attention computation in Transformer models. By proposing a novel convolutional basis and leveraging efficient FFT-based algorithms, we offer new insights into the structure of attention while achieving faster computation with formal guarantees on runtime and approximation error.

As our study is centered on algorithmic and theoretical advancements, supported by controlled experimental validation, we do not anticipate any direct negative societal impacts.