Circuit Complexity Bounds for RoPE-based Transformer Architecture

Bo Chen*   Xiaoyu Li†   Yingyu Liang‡   Jiangxuan Long§   Zhenmei Shi¶

Zhao Song‖

Abstract

Characterizing the express power of the Transformer architecture is critical to understanding its capacity limits and scaling law. Recent works provide the circuit complexity bounds to Transformer-like architecture. On the other hand, Rotary Position Embedding (RoPE) has emerged as a crucial technique in modern large language models, offering superior performance in capturing positional information compared to traditional position embeddings, which shows great potential in application prospects, particularly for the long context scenario. Empirical evidence also suggests that RoPE-based Transformer architectures demonstrate greater generalization capabilities compared to conventional Transformer models. In this work, we establish a circuit complexity bound for Transformers with RoPE attention. Our key contribution is that we show that unless $\mathsf{TC}^0 = \mathsf{NC}^1$, a RoPE-based Transformer with $\mathrm{poly}(n)$-precision, $O(1)$ layers, hidden dimension $d \leq O(n)$ cannot solve the Arithmetic formula evaluation problem or the Boolean formula value problem. This result significantly demonstrates the fundamental limitation of the expressivity of the RoPE-based Transformer architecture, although it achieves giant empirical success. Our theoretical result not only establishes the complexity bound but also may instruct further work on the RoPE-based Transformer.

---

* bc7b@mtmail.mtsu.edu. Middle Tennessee State University.

† xli216@stevens.edu. Stevens Institute of Technology.

‡ yingyul@hku.hk. The University of Hong Kong.   yliang@cs.wisc.edu. University of Wisconsin-Madison.

§ lungchianghsuan@gmail.com. South China University of Technology.

¶ zhmeishi@cs.wisc.edu. University of Wisconsin-Madison.

‖ magic.linuxkde@gmail.com. The Simons Institute for the Theory of Computing at the University of California, Berkeley.

Contents

1   Introduction

Recently, Large Language Models (LLMs), such as GPT-4 [AAA+23], Claude [Ant24], Llama [LT24], and more recently, OpenAI's o1 [Ope24], have exhibited remarkable potential to revolutionize numerous facets of daily life, including conversational AI [LCT+24], AI agents [XCG+23, CYL+24], search capabilities [Ope24], and AI assistants [KHC+24, FJL+24], among others. One of the most significant emergent capabilities of LLMs is their proficiency in handling long-context information, which is essential for effectively processing complex documents such as academic papers, official reports, and legal texts. LLMs also have demonstrated exceptional capabilities in tackling long-context tasks, such as zero-shot summarization [CAM24, ZJV+24] and sustaining coherent, extended conversations [XGW+22, MLT+24]. The o1 model from OpenAI [Ope24] represents a major advancement in this field. By leveraging Chain-of-Thought (CoT) reasoning [WWS+22, KGR+22] and incorporating Retrieval Augmented Generation (RAG) [LPP+20, GXG+23], it showcases a level of expertise comparable to PhD-level problem solving, with both techniques heavily relying on extensive contextual understanding.

Large Language Models (LLMs) are primarily built upon the Transformer architecture [VSP+17], which uses the self-attention mechanism as its core component. Given this foundational structure, an important question arises: what computational primitives can the components of the Transformer implement, and what problems can the entire system solve collectively? These questions are crucial for interpreting Transformers in a principled manner, understanding the potential limitations of their reasoning capabilities, and fostering trust in deployed Transformer-based systems.

To address the aforementioned questions and to investigate the expressiveness of transformers, prior research has made significant strides. Recent studies, such as [MS23b], have established two key results concerning both non-uniform and L-uniform settings: first, any depth-d transformer with $c \log n$-precision can be simulated by a threshold circuit family with constant depth; second, such a transformer can also be simulated by a L-uniform threshold circuit family of constant depth. Further advancing these findings, [MS23a] demonstrate that DLOGTIME-uniform TC^0 circuits are capable of simulating softmax-attention transformers. Building on this foundation, [Chi24] refine these results by increasing the accuracy of approximation. They enhance the precision for softmax-attention transformers from $O(\log n)$ to $O(\mathrm{poly}(n))$, confirming that these transformers fall within the DLOGTIME-uniform TC^0 class. Additionally, they show that a softmax-attention transformer with an absolute error bound of $2^{-O(\mathrm{poly}(n))}$ is also contained within DLOGTIME-uniform TC^0.

On the other hand, first introduced by [SAL+24], Rotary Position Embedding (RoPE) enhances Transformers by encoding both absolute and relative positional information through a rotation matrix, enabling greater sequence length flexibility, improved attention mechanism efficiency, and better performance on long-text tasks, exemplified by RoPE-based language models that can summarize an entire book in a single pass. Due to these advantageous properties, RoPE has been widely adopted in numerous empirical studies [CND+23, BBH+22, BSA+23]. However, despite its considerable success, the underlying mechanisms of RoPE remain largely unknown, posing an intriguing mystery in the field. A natural question arises:

Does RoPE *enhance the expressiveness of the Transformer-based Large Language Model?*

This work aims to address this question from the perspective of circuit complexity, taking a step forward in theoretically understanding the underlying mechanisms of RoPE. We present a rigorous theoretical investigation of RoPE-based Transformers, establishing fundamental limits on their computational power.

Our core approach involved a systematic examination of the circuit complexity for each component of the RoPE-based architecture, from the basic trigonometric functions to the complete

attention mechanism. Ultimately, we prove that these models can be simulated using uniform TC^0 circuits. Furthermore, we show that unless $\mathsf{TC}^0 = \mathsf{NC}^1$, RoPE-based Transformers with $\mathrm{poly}(n)$-precision, $O(1)$ layers, and a hidden dimension $d \leq O(n)$ are unable to solve either Arithmetic formula evaluation or Boolean formula value problems. This finding is significant because it uncovers fundamental expressivity limitations of RoPE-based architectures, even though they have shown empirical success in modern language models.

Beyond Merrill and Sabharwal [MS23b, MS23a] and Chiang [Chi24], our contribution are summarized as follows:

- We prove that unless $\mathsf{TC}^0 = \mathsf{NC}^1$, RoPE-based Transformer with $\mathrm{poly}(n)$-precision, constant-depth, $\mathrm{poly}(n)$-size can be simulated by a DLOGTIME-uniform TC^0 circuit family (Theorem 4.8).

- We prove that unless $\mathsf{TC}^0 = \mathsf{NC}^1$, a RoPE-based Transformer with $\mathrm{poly}(n)$-precision, $O(1)$ layers, hidden dimension $d \leq O(n)$ cannot solve the Arithmetic formula evaluation problems (Theorem 5.8).

- We prove that unless $\mathsf{TC}^0 = \mathsf{NC}^1$, a RoPE-based Transformer with $\mathrm{poly}(n)$-precision, $O(1)$ layers, hidden dimension $d \leq O(n)$ cannot solve the Boolean formula value problem (Theorem 5.9).

Roadmap. In Section 2, we review the related work. In Section 3, we introduce some important computation concepts and Transformer definitions essential for the subsequent sections. In Section 4, we give the circuit complexity result of RoPE-based Transformers. In Section 5, we give our hardness results. In Section 6, we summarizes our theoritical results.

2 Related work

Complexity and Neural Networks. Circuit complexity, a branch of computational complexity theory, studies circuit families as models of computation. Several circuit complexity classes are significant in machine learning. Specifically, AC^0 represents problems highly parallelizable with standard logic gates, while TC^0 extends this to include *threshold gates*, and NC^1 denotes the language recognizable by $O(\log n)$-depth circuits with bounded gate arity [MSS22]. It is known that $\mathsf{AC}^0 \subset \mathsf{TC}^0 \subseteq \mathsf{NC}^1$, but whether $\mathsf{TC}^0 \neq \mathsf{NC}^1$ remains an open question. Assuming this inequality, [LAG$^+$22] shows that Transformer depth must depend on input sequence length when simulating non-solvable semiautomata. [LLZM24] explore relationships among constant-depth Transformers, Transformers with Chain-of-Thought (CoT), and circuit complexity. They demonstrate: $\mathsf{T}[\mathrm{poly}(n), 1, 1] \subseteq \mathsf{CoT}[\log n, \mathrm{poly}(n), 1, 1] \subseteq \mathsf{AC}^0$ and $\mathsf{T}[\mathrm{poly}(n), \log n, 0] \subseteq \mathsf{CoT}[\log n, \mathrm{poly}(n), \log n, 0] \subseteq \mathsf{TC}^0$ where $\mathsf{T}[d(n), s(n), e(n)]$ denotes a constant-depth Transformers with embedding size $d(n)$, precision $s(n)$ bits, and exponent bits $e(n)$ for input length n and $\mathsf{CoT}[T(n), d(n), s(n), e(n)]$ denotes a $T(n)$-step CoT of a constant-depth Transformer $\mathsf{T}[d(n), s(n), e(n)]$. Their results provide theoretical insights into the emergent CoT ability of Transformers, showing that intermediate reasoning steps enable tackling more complex problems. The Strong Exponential Time Hypothesis (SETH), introduced by [IP01], strengthens the $\mathsf{P} \neq \mathsf{NP}$ conjecture by asserting that current best SAT algorithms are roughly optimal: for every $\epsilon > 0$, there exists $k \geq 3$ such that k-SAT cannot be solved in $O(2^{(1-\epsilon)n})$ time, even randomly. SETH is widely used to prove fine-grained lower bounds for various algorithmic problems [Wil18] and has been applied to derive lower bounds for Transformer training/inference [AS23a, AS24b, LSS$^+$24c] and tensor attention [AS23b, LSSZ24b]. Specifically,

[AS23a] demonstrates that unless the SETH fails, no algorithm exists that can compute the forward pass of an attention network in truly-subquadratic time. On the other hand, [AS24b] establishes that the same condition applies to the backward computation of attention networks, i.e., unless the SETH fails, no truly-subquadratic time algorithm can be devised for the backward computation of attention networks. In essence, complexity theory provides a powerful framework for investigating the computational capabilities of neural networks, by rigorously analyzing the computational problems they can efficiently solve.

Limitations of Transformers. Transformers have shown exceptional capabilities in natural language processing tasks, yet their effectiveness in mathematical computations remains limited [Cha22]. Consequently, research efforts have increasingly focused on defining the computational boundaries of Transformers. These studies investigate two types of Transformers: (1) the average-head attention Transformer, where the largest entry in the probability vector is set to 1 and all other entries are set to 0; (2) the softmax-attention Transformer, where the probability vector is produced using a softmax function, formally defined as $\mathsf{Softmax}(X) = \mathrm{diag}(\exp(X) \cdot \mathbf{1}_n)^{-1} \cdot \exp(X)$. For the average-head attention Transformer, Merrill, Sabharwal, and Smith [MSS22] demonstrate that it can recognize languages beyond the circuit complexity class AC^0 but can be simulated by constant-depth threshold circuits, placing it within the non-uniform TC^0 class. Additionally, [LAG+22] prove that softmax-attention Transformers can be simulated by a non-uniform TC^0 circuit. Extending this analysis, [MS23b] introduce a generalized similarity function $s : \{0,1\}^p \times \{0,1\}^p \to \{0,1\}^p$, applicable to any similarity function within this mapping, and show that softmax-attention Transformers belong to L-uniform TC^0. Through the conversion of Transformer operations into sentences in FOM (first-order logic extended to include MAJORITY quantifiers [Imm98]), [MS23a] demonstrate that DLOGTIME-uniform TC^0 can simulate softmax-attention Transformers. [Chi24] further refine these findings by enhancing approximation accuracy. Specifically, they eliminate error in average-head attention Transformers and improve the precision for softmax-attention Transformers from $O(\log n)$ to $O(\mathrm{poly}(n))$, proving that these Transformers belong to the DLOGTIME-uniform TC^0 class. Additionally, they show that a softmax-attention Transformer with an absolute error of at most $2^{-O(\mathrm{poly}(n))}$ is also within DLOGTIME-uniform TC^0. Regarding more practical tasks such as mathematical and decision-making problems, [FZG+23] show that, unless $\mathsf{TC}^0 = \mathsf{NC}^1$, no log-precision Transformer can solve arithmetic and equation-solving problems, nor can any log-precision autoregressive Transformer generate correct answers for the Context-Free Grammar (CFG) Membership Testing problem [Sip96]. These theoretical constraints help explain some of the practical limitations observed when applying Transformers to mathematical tasks. Much of the relevant work in recent years has been related to this such as looped transformer [AS24a, LSS+24a, LSS+24b, CLS+24], acceleration [HYW+23, CLL+24, LLS+24e, LSSY24, LLSS24, LLS+24d, SMN+24, LLS+24c, LLS+24b, HWL+24b, HCL+24, HLSL24, WHL+24, XHH+24, HCW+24, SZZ24, WHHL24, HWL24a], graph attention [VCC+17, WJS+19, BAY21, CHL+24] and other related works[DSWY22, SY23, SSX23, GMS23, XSL24, LLS+24a, LSSZ24a, HSK+24].

3 Preliminary

In this section, we present some preliminary concepts and definitions of our paper. In Section 3.1, we introduce some basic notations used in our paper. In Section 3.2, we introduce the basics of the circuit complexity classes. In Section 3.3, we state the Boolean formula value problem and Arithmetic formula evaluation problem and define some important tools to set up our problem. In Section 3.4, we introduce Rotary Position Embedding (RoPE) attention and some basic settings in

the Transformer.

3.1 Notations

For any positive integer n, we use $[n]$ to denote set $\{1, 2, \cdots, n\}$. We use $\mathbb{N} := \{0, 1, 2, \ldots\}$ to denote the set of natural numbers. We use $\Pr[]$, $\mathbb{E}[]$, and $\mathrm{Var}[]$ to denote the probability, expectation, and variance, respectively. For two vectors $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^n$, we use $\langle x, y \rangle$ to denote the inner product between x, y. We use $\mathbf{1}_n$ to denote a length-n vector where all the entries are ones. We use $X_{i,j}$ to denote the i-th row, j-th column of $X \in \mathbb{R}^{m \times n}$. We use $\|A\|_\infty$ to denote the ℓ_∞ norm of a matrix $A \in \mathbb{R}^{n \times d}$, i.e. $\|A\|_\infty := \max_{i \in [n], j \in [d]} |A_{i,j}|$. For $x_i \in \{0, 1\}^*$, x_i is a binary number of arbitrary length, more generally speaking, x_i is a binary string of length p, where each bit is either 0 or 1.

3.2 Circuit Complexity

The Boolean circuit, using AND, OR, and NOT gates, is a fundamental computational model in computer science, which is formally defined as follows.

Definition 3.1 (Boolean circuit, Definition 6.1 on page 102 of [AB09]). *A Boolean circuit with n variables is a function $C_n : \{0, 1\}^n \to \{0, 1\}$ defined on a directed acyclic graph. The nodes in this graph represent logic gates such as AND, OR, and NOT. Input nodes, which have an in-degree of 0, are assigned one of the n Boolean variables. The circuit evaluates each non-input gate's value by computing the inputs it receives from other gates.*

It is natural to examine the languages that can be recognized by specific families of Boolean circuits since it offers insights into the computational capabilities and efficiency of a certain family of computational models.

Definition 3.2 (Languages recognized by a circuit family, Definition 6.2 on page 103 of [AB09]). *We say that a language $L \subseteq \{0, 1\}^*$ is recognized by a family $\mathcal{C}$ of Boolean circuits if for all $x \in \{0, 1\}^*$, there exists a Boolean circuit $C_{|x|} \in \mathcal{C}$ over $|x|$ variables such that $C_{|x|}(x) = 1$ if and only if $x \in L$.*

We now define classes of languages by imposing constraints on the types of logic gates that can be utilized within the circuit families necessary for their recognition. The weakest one we are going to introduce is the NC^i class.

Definition 3.3 (NC^i, Definition 6.21 on page 109 of [AB09]). *The class NC^i consists of languages that can be recognized by Boolean circuits with $O(\mathrm{poly}(n))$ size, $O((\log n)^i)$ depth, and bounded fan-in AND, OR gates, and NOT gates.*

When Boolean circuits permit AND and OR gates with unbounded fan-in, they gain the capacity to recognize a large class of languages. We define AC^i class as follows.

Definition 3.4 (AC^i, Definition 6.22 on page 109 of [AB09]). *The class AC^i consists of languages that can be recognized by Boolean circuits with $O(\mathrm{poly}(n))$ size, $O((\log n)^i)$ depth, and unbounded fan-in AND, OR gates, and NOT gates.*

In fact, AND, OR gates, and NOT gates can all be implemented by MAJORITY gates, where the MAJORITY gate outputs 0 when half or more arguments are 0 and outputs 1 otherwise. Thus, if we allow Boolean circuits to be equipped with MAJORITY gates, we get a larger class TC^i.

Definition 3.5 (TC^i, Definition 4.34 on page 126 of [Vol99]). *The class TC^i consists of languages that can be recognized by Boolean circuits with $O(\mathrm{poly}(n))$ size, $O((\log n)^i)$ depth, and unbounded fan-in* AND, OR *gates,* NOT *gates, and* MAJORITY *gates which can output 1 when more than half of their inputs are 1.*

Remark 3.6. *Alternatively, in Definition 3.5,* MAJORITY *gates can be replaced by* THRESHOLD *or* MOD *gates configured around prime values. When a Boolean circuit equipped with any of them, we call it is a threshold circuit.*

Finally, we recall the definition of P class.

Definition 3.7 (P, Definition 1.20 on page 9 of [AB09]). *The class* P *consists of languages that can be recognized by a deterministic Turing machine in polynomial time in input size.*

The following fact is a folklore that gives the hierarchy of circuit families.

Fact 3.8 (Folklore, page 110 on [AB09], Corollary 4.35 on page 126 of [Vol99]). *For all $i \in \mathbb{N}$,* $\mathsf{NC}^i \subseteq \mathsf{AC}^i \subseteq \mathsf{TC}^i \subseteq \mathsf{NC}^{i+1} \subseteq \mathsf{P}$.

Remark 3.9. *For $i = 0$, it is known that $\mathsf{NC}^0 \subsetneq \mathsf{AC}^0 \subsetneq \mathsf{TC}^0$. However, whether $\mathsf{TC}^0 \subsetneq \mathsf{NC}^1$ is an open problem in circuit complexity. Whether $\mathsf{NC} := \cup_{i \in \mathbb{N}} \mathsf{NC}^i \subsetneq \mathsf{P}$ is also an open problem. See page 110 in [AB09], page 116 in [Vol99] for discussion about these.*

We have defined non-uniform circuit families, which do not need to share structure across varying input sizes and can theoretically handle undecidable problems but are impractical due to their infinite description length. Uniform circuit families offer a more feasible computational model, relevant to complexity and language theory. We first define L-uniformity as follows.

Definition 3.10 (L-uniformity, Definition 6.5 on page 104 of [AB09]). *Let* C *be a language recognized by a circuit family $\mathcal{C}$ (e.g.* C *can be $\mathsf{NC}^i, \mathsf{AC}^i$, or TC^i). We say that a language $L \subseteq \{0, 1\}^*$ is in* L*-uniform* C *if there exists a Turing machine that, for every $n \in \mathbb{N}$, maps 1^n to a circuit in $\mathcal{C}$ over n variables using $O(\log n)$ space such that C_n recognizes L.*

Next, we define DLOGTIME-uniformity and remark on the relationship between these two different uniformity definitions.

Definition 3.11 (DLOGTIME-uniformity, Definition 4.28 on page 123 of [BI94]). *Let* C *be a language recognized by a circuit family $\mathcal{C}$ (e.g.* C *can be $\mathsf{NC}^i, \mathsf{AC}^i$, or TC^i). We say that a language $L \subseteq \{0, 1\}^*$ is in* DLOGTIME*-uniform* C *if there exists a random access Turing machine that, for every $n \in \mathbb{N}$, maps 1^n to a circuit C_n over n variables in $\mathcal{C}$ in $O(\log n)$ time such that C_n recognizes L.*

Remark 3.12. DLOGTIME*-uniformity is equivalent to* L*-uniformity, with the exception of small circuit complexity classes where the circuits lack the capacity to simulate the machines that create them. See [BI94, HAB02] for more discussion on different notions of uniformity. In this paper, whenever we refer to uniform TC^0, we specifically mean* DLOGTIME*-uniform TC^0.*

3.3 Float Point Numbers

In this section, we introduce some important definitions. To establish a foundation for our computational framework, we first introduce the essential definitions of floating-point numbers and their operations, which are crucial for implementing Transformer calculations efficiently.

Definition 3.13 (Floating-point number, Definition 9 on Page 5 of [Chi24]). *A p-bit floating-point number is a pair $\langle m, e \rangle$ of two integers where the significance $m \in (-2^p, -2^{p-1}] \cup \{0\} \cup [2^{p-1}, 2^p)$ and the exponent $e \in [-2^p, 2^p)$. The value of the floating point $\langle m, e \rangle$ is the real number $m \cdot 2^e$. We denote the set of all p-bits floating-point numbers as $\mathbb{F}_p$.*

To handle these floating-point numbers in practice, we need precise rules for rounding and basic arithmetic operations:

Definition 3.14 (Rounding, Definition 9 on page 5 of [Chi24]). *Let x be a real number or a floating point. We define $\mathrm{round}_p(x)$ as the p-bit floating-point number nearest to x. When there are two such numbers, we define $\mathrm{round}_p(x)$ as the one with even significance.*

Building on these fundamental definitions, we can now define the core arithmetic operations needed for Transformer computations:

Definition 3.15 (Floating-point number operations, page 5 on [Chi24]). *Let a, b be two integers, we define*

$$a \mathbin{/\!/} b := \begin{cases} a/b & \text{if } a/b \text{ is a mutiple of } 1/4, \\ a/b + 1/8 & \text{otherwise.} \end{cases}$$

Given two p-bits floating points $\langle m_1, e_1 \rangle, \langle m_2, e_2 \rangle$, we define the following operations:

- *addition:*

$$\langle m_1, e_1 \rangle + \langle m_2, e_2 \rangle := \begin{cases} \mathrm{round}_p(\langle m_1 + m_2 \mathbin{/\!/} 2^{e_1 - e_2}, e_1 \rangle) & \text{if } e_1 \geq e_2, \\ \mathrm{round}_p(\langle m_1 \mathbin{/\!/} 2^{e_2 - e_1} + m_2, e_2 \rangle) & \text{if } e_1 \leq e_2. \end{cases}$$

- *multiplication:*

$$\langle m_1, e_1 \rangle \times \langle m_2, e_2 \rangle := \mathrm{round}_p(\langle m_1 m_2, e_1 + e_2 \rangle).$$

- *division:*

$$\langle m_1, e_1 \rangle \div \langle m_2, e_2 \rangle := \mathrm{round}_p(\langle m_1 2^{p-1} \mathbin{/\!/} m_2, e_1 - e_2 - p + 1 \rangle).$$

- *comparison:*

$$\langle m_1, e_1 \rangle \leq \langle m_2, e_2 \rangle \Leftrightarrow \begin{cases} m_1 \leq m_2 \mathbin{/\!/} 2^{e_1 - e_2} & \text{if } e_1 \geq e_2, \\ m_1 \mathbin{/\!/} 2^{e_2 - e_1} \leq m_2 & \text{if } e_1 \leq e_2. \end{cases}$$

- *floor:*

$$\lfloor \langle m, e \rangle \rfloor := \begin{cases} \langle m 2^e, 0 \rangle & \text{if } e \geq 0, \\ \mathrm{round}(\langle m/2^{-e}, 0 \rangle) & \text{if } e < 0. \end{cases}$$

These operations are not just theoretical constructs, they can be efficiently implemented in hardware, as demonstrated by the following lemma:

Lemma 3.16 (Standard float point number operations in TC^0, Lemma 10 on page 5 and Lemma 11 on page 6 of [Chi24]). *Let p be a positive integer. If $p \leq \mathrm{poly}(n)$, then the following statements hold:*

- *Part 1. The addition, multiplication, division, and comparison defined in Definition 3.15 of two p-bit floating point numbers is computable by a constant-depth uniform threshold circuit of size $\mathrm{poly}(n)$. We use d_{std} to denote the maximum depth needed for these operations.*

- *Part 2. The iterated multiplication of n p-bit floating point numbers is computable by a constant-depth uniform threshold circuit of size $\mathrm{poly}(n)$. We use $d_{\otimes}$ denote the depth needed for the iterated multiplication.*

- *Part 3. The iterated addition of n p-bit floating point numbers (rounding after the summation is completed) is computable by a constant-depth uniform threshold circuit of size $\mathrm{poly}(n)$. We use $d_{\oplus}$ denote the depth needed for the iterated addition.*

Corollary 3.17 (Floor operation in TC^0). *Let p be a positive integer. If $p \leq \mathrm{poly}(n)$, then floor operation defined in Definition 3.15 of a p-bit floating point number is computable by a constant-depth uniform threshold circuit of size $\mathrm{poly}(n)$. The maximum depth needed for floor operations is bounded by d_{std} in Lemma 3.16.*

Proof. This directly follows from the definition of the floor function in Definition 3.15. $\square$

Lemma 3.18 (Approximating exp in TC^0, Lemma 12 on page 7 of [Chi24]). *If a positive integer $p \leq \mathrm{poly}(n)$, then for every p-bit floating point number x, there is a constant-depth uniform threshold circuit of size $\mathrm{poly}(n)$ which can compute $\exp(x)$ with a relative error at most 2^{-p}. We use $d_{\exp}$ to denote the depth needed for computing $\exp(x)$.*

Lemma 3.19 (Approximating square root in TC^0, Lemma 12 on page 7 of [Chi24]). *If a positive integer $p \leq \mathrm{poly}(n)$, then for every p-bit floating point number x, there is a constant-depth uniform threshold circuit of size $\mathrm{poly}(n)$ which can compute $\sqrt{x}$ with a relative error at most 2^{-p}. We use d_{sqrt} to denote the depth needed for computing $\sqrt{x}$.*

3.4 Transformer Blocks

With our mathematical foundation established, In this section, we can now describe the key components of Transformer architecture, beginning with the softmax operation that is fundamental to attention mechanisms.

Definition 3.20 (Softmax). *Let $z \in \mathbb{F}_p^n$. We define $\mathsf{Softmax} : \mathbb{F}_p^n \to \mathbb{F}_p^n$ satisfying*

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

A key innovation in modern Transformers is the RoPE, which begins with a basic rotation matrix:

Definition 3.21 (Rotation matrix block). *Let n be the input sequence length, d is given the embedding dimension, $\theta \in \mathbb{F}_p$, we define the rotation matrix as*

$$R(\theta) := \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}.$$

This basic rotation matrix is then extended to handle relative positions in the sequence.

Definition 3.22 (Rotation matrix). *Let j be the index of position in the sequence, i the index of tokens, we define the overall relative rotation matrix*

$$R_{j-i} = \begin{bmatrix} R((j-i)\theta_1) & 0 & \cdots & 0 \\ 0 & R((j-i)\theta_2) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & R((j-i)\theta_{d/2}) \end{bmatrix}.$$

where the angle frequencies $\theta_1, \cdots, \theta_{d/2}$ are a set of given parameters, for details on specifying θ, see Equation (15) on page 5 of [SAL+24].

Using these rotation matrices, we can define the RoPE attention mechanism, which incorporates positional information directly into the attention computation.

Definition 3.23 (RoPE attention matrix). *Let Rotation matrix R_{j-i} be defined in Definition 3.22. Let $W_Q, W_K \in \mathbb{F}_p^{d \times d}$ denote the model weights. Let $X \in \mathbb{F}_p^{n \times d}$ denote the representation of the length-n sentence. Then, we define the new attention matrix $A \in \mathbb{F}_p^{n \times n}$ by, For $i, j \in [n]$,*

$$A_{i,j} := \exp(\underbrace{X_{i,*}}_{1 \times d} \underbrace{W_Q}_{d \times d} \underbrace{R_{j-i}}_{d \times d} \underbrace{W_K^\top}_{d \times d} \underbrace{X_{j,*}^\top}_{d \times 1}).$$

The attention matrix is then used to compute a single attention layer.

Definition 3.24 (Single attention layer). *Let $X \in \mathbb{F}_p^{n \times d}$ denote the representation of the length-n sentence. Let $W_V \in \mathbb{F}_p^{d \times d}$ denote the model weights. As in the usual attention mechanism, the final goal is to output an $n \times d$ size matrix where $D := \mathrm{diag}(A\mathbf{1}_n) \in \mathbb{F}_p^{n \times n}$. Then, we define the i-th attention layer Attn as*

$$\mathsf{Attn}_i(X) := D^{-1} A X W_V.$$

We combine multiple attention layers with other components to create a complete Transformer architecture.

Definition 3.25 (Multi-layer RoPE-based Transformer). *Let m denote the number of Transformer layers in the model. Let g_i denote components other than self-attention in the i-th Transformer layer, where $g_i : \mathbb{F}_p^{n \times d} \to \mathbb{F}_p^{n \times d}$ for any $i \in \{0, 1, 2, \ldots, m\}$. Let Attn_i denote the self-attention module in the i-th Transformer layer (see also Definition 3.24). Let $X \in \mathbb{F}_p^{n \times d}$ denote the input data matrix. We define a m-layer Transformer $\mathsf{TF} : \mathbb{F}_p^{n \times d} \to \mathbb{F}_p^{n \times d}$ as*

$$\mathsf{TF}(X) := g_m \circ \mathsf{Attn}_m \circ \cdots \circ g_1 \circ \mathsf{Attn}_1 \circ g_0(X) \quad \in \mathbb{F}_p^{n \times d},$$

where $\circ$ denotes function composition.

Here we introduce two different kinds of g_i function. First, we introduce the MLP (Multilayer Perceptron) layer.

Definition 3.26 (Multilayer Perceptron layer). *Let $X \in \mathbb{F}_p^{n \times d}$ denote the input data matrix. Let $i \in [n]$. Then, we define the MLP layer as follows:*

$$g^{\mathrm{MLP}}(X)_{i,*} := \underbrace{W}_{d \times d} \cdot \underbrace{X_{i,*}}_{d \times 1} + \underbrace{b}_{d \times 1}.$$

Then, we introduce the LN (Layer-wise Normalization) layer:

Definition 3.27 (Layer-wise normalization layer). *Let $X \in \mathbb{F}_p^{n \times d}$ denote the input data matrix. Let $i \in [n]$. Then, we define the LN layer as follows:*

$$g^{\mathrm{LN}}(X)_{i,*} := \frac{X_{i,*} - \mu_i}{\sqrt{\sigma_i^2}},$$

where $\mu_i := \sum_{j=1}^{d} X_{i,j}/d$, and $\sigma_i^2 := \sum_{j=1}^{d}(X_{i,j} - \mu_i)^2/d$.

This multi-layer architecture forms the backbone of modern Transformer models, combining the floating-point operations, attention mechanisms, and positional embeddings defined above into a powerful sequence processing system.

4 Complexity of RoPE-based Transformers

In this section, we establish several fundamental results regarding the circuit complexity of basic operations required in Transformer computations. In Section 4.1, we begin by analyzing trigonometric functions, which are essential for rotary position embeddings. In Section 4.2, we then proceed to study matrix operations. In Section 4.3, we examine the RoPE-based attention matrix. In Section 4.4, we analyze the single RoPE-Attention layer. In Section 4.5, we compute some common components other than the self-attention layer. In Section 4.6, we show more details about the complete RoPE-based Transformer mechanism. In Section 4.7, we show our main results that the circuit complexity bound of RoPE-based Transformer.

4.1 Approximating Trigonometric Functions

In this section, we first demonstrate that basic trigonometric functions, i.e., sine and cosine function, which are fundamental to RoPE embeddings, can be computed by threshold circuits.

Lemma 4.1 (Trigonometric function approximation in TC^0, informal version of Theorem A.1 at Appendix A). *If $p \leq \mathrm{poly}(n)$, then for every p-bit floating point number x, there is a constant-depth uniform threshold circuit of size $\mathrm{poly}(n)$ and depth $8d_{\mathrm{std}} + d_\oplus + d_\otimes$ which can compute $\sin(x)$ and $\cos(x)$ with a relative error at most 2^{-p}. To simplify, we use $d_\triangle$ denote the depth needed for computing $\sin(x)$ and $\cos(x)$.*

4.2 Computing Matrix Products

In this section, we show that basic matrix multiplication can be computed in TC^0.

Lemma 4.2 (Matrix multiplication in TC^0, informal version of Lemma B.1 at Appendix B). *Let $A \in \mathbb{F}_p^{n_1 \times d}$, $B \in \mathbb{F}_p^{d \times n_2}$ be two matrices. If $p \leq \mathrm{poly}(n), n_1, n_2 \leq \mathrm{poly}(n), d \leq n$, then AB can be computable by a uniform threshold circuit with size $\mathrm{poly}(n)$ and depth $(d_{\mathrm{std}} + d_\oplus)$.*

4.3 Computing RoPE-based Attention Matrix

In this section, we extend this to the computation of the attention matrix with positional embeddings, i.e., RoPE-based attention matrix computation.

Lemma 4.3 (RoPE-based attention matrix computation in TC^0, informal version of Lemma B.2 at Appendix B). *If $p \leq \mathrm{poly}(n)$, then the attention matrix A in Definition 3.23 can be computable by a uniform threshold circuit with size $\mathrm{poly}(n)$ and depth $4(d_{\mathrm{std}} + d_\oplus) + d_\triangle + d_{\exp}$.*

4.4 Computing Single RoPE-based Attention Layer

In this section, we analyze the complete attention layer, the approach allows us to carefully track the circuit depth requirements at each stage.

Lemma 4.4 (Single RoPE-based attention layer computation in TC^0). *If $p \leq \mathrm{poly}(n)$, then the attention layer* Attn *in Definition 3.24 can be computable by a uniform threshold circuit with size* $\mathrm{poly}(n)$ *and depth* $7(d_{\mathrm{std}} + d_{\oplus}) + d_{\triangle} + d_{\exp}$.

Proof. To compute Attn, we need to multiply 4 matrix, namely D^{-1}, A, X and W_V. To get these matrices, we need to compute D and A. following from $D := \mathrm{diag}(A\mathbf{1}_n)$, D can be computed by a depth $d_{\oplus}$, size $\mathrm{poly}(n)$ uniform threshold circuit following from Part 3. of Lemma 3.16. Following from Lemma 4.3, computing A needs a circuit of depth $4(d_{\mathrm{std}} + d_{\oplus}) + d_{\triangle} + d_{\exp}$. Then, we can multiply A, X and W_V, which can be computed by a depth $2(d_{\mathrm{std}} + d_{\oplus})$, size $\mathrm{poly}(n)$ uniform threshold circuit following from Lemma 4.2. Finally, we can compute $D^{-1} \cdot AXW_V$ by apply division in parallel, which can be computed by a depth d_{std}, size $\mathrm{poly}(n)$ uniform threshold circuit following from Part 1. of Lemma 3.16. Combining above circuit, we have

$$d_{\mathrm{total}} = 7(d_{\mathrm{std}} + d_{\oplus}) + d_{\triangle} + d_{\exp}.$$

Because the number of parallel operation are $O(\mathrm{poly}(n))$, we can show that $\mathsf{Attn}(X)$ can be computed by a depth $7(d_{\mathrm{std}} + d_{\oplus}) + d_{\triangle} + d_{\exp}$, size $\mathrm{poly}(n)$ uniform threshold circuit.

Thus we complete the proof. $\square$

4.5 Computing Common Buliding Blocks other than Self-attention layer

In Definition 3.25, we define Multi-layer RoPE-based Transformer with self-attention layer and other components, for example layer-norm and MLP. In this section, we show how to compute these components. We first give the circuit complexity for the MLP layer.

Lemma 4.5 (MLP computation in TC^0, informal version of Lemma B.3 at Appendix B). *If $p \leq \mathrm{poly}(n)$, then the MLP layer in Definition 3.26 can be computable by a uniform threshold circuit with size* $\mathrm{poly}(n)$ *and depth* $2d_{\mathrm{std}} + d_{\oplus}$.

Then, we give the circuit complexity for the layer-normalization layer.

Lemma 4.6 (Layer-norm computation in TC^0, informal version of Lemma B.4 at Appendix B). *If $p \leq \mathrm{poly}(n)$, then the Layer-wise Normalization layer in Definition 3.27 can be computable by a uniform threshold circuit with size* $\mathrm{poly}(n)$ *and depth* $5d_{\mathrm{std}} + 2d_{\oplus} + d_{\mathrm{sqrt}}$.

4.6 Computing Multi-layer RoPE-based Transformer

In this section, we show how to compute the multi-layer RoPE-Transformer.

Lemma 4.7 (Multi-layer RoPE-based Transformer computation in TC^0, informal version of Lemma B.5 at Appendix B). *Suppose that for each $i \in [m]$, g_i in* TF *is computable by a constant depth d_g uniform threshold circuit with size $\mathrm{poly}(n)$. If $p \leq \mathrm{poly}(n)$, then the* RoPE-*based Transformer* TF *in Definition 3.25 can be computable by a uniform threshold circuit with size* $\mathrm{poly}(n)$ *and depth* $(m+1)d_g + 7m(d_{\mathrm{std}} + d_{\oplus}) + m(d_{\triangle} + d_{\exp})$.

4.7 Main Result: Circuit Complexity Bound of RoPE-based Transformers

In this section, we are ready to represent our main result. We show the circuit complexity bound of RoPE-based Transformer.

Theorem 4.8 (Main result, Circuit complexity bound of RoPE-based Transformers). *Suppose that for each $i \in [m]$, g_i in TF is computable by a constant depth d_g uniform threshold circuit with size $\mathrm{poly}(n)$. If $p \leq \mathrm{poly}(n), d \leq O(n), m \leq O(1)$, then the RoPE-based Transformer TF in Definition 3.25 can be simulated by a uniform TC^0 circuit family.*

Proof. Since $m = O(1)$, by Lemma 4.7, the circuit that computes $\mathsf{TF}(X)$ has depth

$$(m+1)d_g + 7m(d_{\mathrm{std}} + d_\oplus) + m(d_\triangle + d_{\exp}) = O(1)$$

and size $\mathrm{poly}(n)$. Therefore it can be simulated by a uniform TC^0 circuit family.

Thus we complete the proof. □

In Theorem 4.8, we prove that unless $\mathsf{TC}^0 = \mathsf{NC}^1$, RoPE-based Transformer with $\mathrm{poly}(n)$-precision, constant-depth, $\mathrm{poly}(n)$-size can be simulated by a DLOGTIME-uniform TC^0 circuit family. It means that although the RoPE-based Transformers gain success empirically, it still suffers fundamental expressivity limitations under circuit complexity. We introduce these limitations in the following section.

5 Hardness

In this section, we state two important problems and the corresponding hardness results. In Section 5.1, we introduce the Arithmetic formula evaluation problem. In Section 5.2, we introduce the Boolean formula value problem. In Section 5.3, we state our two hardness results.

5.1 Arithmetic Formula Evaluation Problem

In this section, we first provide a foundational definition as established in [BCGR92].

Definition 5.1 (Arithmetic formula, Definition on page 13 of [BCGR92]). *Let $\mathbb{S}$ be a semi-ring (which may also be a ring or field). An arithmetic formula over $\mathbb{S}$ with indeterminates $X_1, X_2, \cdots, X_n$ is defined by:*

- *For $i \in [n]$, X_i is an arithmetic formula.*

- *For every $c \in \mathbb{S}$, c is an arithmetic formula.*

- *If α is an arithmetic formula and θ is a unary operator of $\mathbb{S}$ then $(\theta\alpha)$ is arithmetic formula.*

- *If α and β are arithmetic formulas and θ is a binary operator of $\mathbb{S}$ then $(\alpha\theta\beta)$ is an arithmetic formula.*

An arithmetic formula A with indeterminates $X_1, \cdots, X_n$ is denoted by $A(X_1, \cdots, X_n)$.

Following the definition, we explore its computational implications.

Definition 5.2 (Arithmetic formula evaluation problem, Definition on page 14 of [BCGR92]). *Let $\mathbb{S}$ be a ring, field, or semi-ring. The arithmetic formula evaluation problem is: Given an arithmetic formula $A(X_1, X_2, \cdots, X_n)$ over $\mathbb{S}$ and constants $c_1, c_2, \cdots, c_n \in \mathbb{S}$, what is $A(c_1, c_2, \cdots, c_n)$?*

12

Building upon the previously established definitions, we then establish the computational complexity of the problem.

Lemma 5.3 (Theorem 6.1 on page 31 of [BCGR92]). *The arithmetic formula evaluation problem is in* NC^1-*complete.*

5.2 Boolean Formula Value Problem

In this section, we now shift our focus to the domain of Boolean formulas and their evaluation.

Definition 5.4 (Definition on Page 1 of [Bus87]). *Let* $\Sigma = \{0, 1, \wedge, \vee, \neg, (,)\}$, *the Boolean formula are given by the following inductive definition:*

- *0 and 1 are Boolean formulas.*

- *If* α *and* β *are Boolean formulas, then so are* $(\neg\alpha), (\alpha \wedge \beta)$ *and* $(\alpha \vee \beta)$.

To detail further attributes of these formulas:

Definition 5.5 (Definition on page 1 of [Bus87]). $|\alpha|$ *is the length of* α, *i.e. the number of symbols in the string* α.

Definition 5.6 (Definition on page 1 of [Bus87]). *The Boolean formula is defined by the following inductive definition:*

- *0 and 1 are Boolean formulas.*

- *If* α *is a Boolean formula then so is* $\alpha\neg$.

- *If* α *and* β *are Boolean formulas and if* $|\alpha| \geq |\beta|$ *then* $\alpha\beta\vee$ *and* $\alpha\beta\wedge$ *are Boolean formulas.*

The Boolean formula is defined in the usual way, where 0 and 1 represent False and True, respectively.

Lemma 5.7 (Page 1 on [Bus87]). *The problem of determining the truth value of a Boolean formula is in* NC^1-*complete.*

5.3 Hardness Results

In this section, we state our two hardness results.

Theorem 5.8. *Unless* $\mathsf{TC}^0 = \mathsf{NC}^1$, *a* RoPE-*based Transformer with* $\mathrm{poly}(n)$-*precision,* $O(1)$ *layers, hidden dimension* $d \leq O(n)$ *cannot solve the Arithmetic formula evaluation problems.*

Proof. This follows from combining Theorem 4.8 (circuit complexity bound of RoPE-base Transformer) and Lemma 5.3 (the arithmetic formula evaluation problem is in NC^1) which we proved above, and Fact 3.8 (hierarchy of circuit families). Thus we complete the proof. □

Theorem 5.9. *Unless* $\mathsf{TC}^0 = \mathsf{NC}^1$, *a* RoPE-*based Transformer with* $\mathrm{poly}(n)$-*precision,* $O(1)$ *layers, hidden dimension* $d \leq O(n)$ *cannot solve the Boolean formula value problem.*

Proof. This follows from combining Theorem 4.8 (circuit complexity bound of RoPE-base Transformer) and Lemma 5.7 (the problem of determining the truth value of a Boolean formula is in NC^1) which we proved above, and Fact 3.8 (hierarchy of circuit families). Thus we complete the proof. □

6 Conclusion

In this work, we provide a rigorous theoretical analysis of RoPE-based Transformers, establishing fundamental bounds on their computational capabilities. Our main idea was to systematically analyze the circuit complexity of each component in the RoPE-based architecture, from basic trigonometric functions to the complete attention mechanism, ultimately proving that these models can be simulated by uniform TC^0 circuits. More importantly, we demonstrate that unless $\mathsf{TC}^0 = \mathsf{NC}^1$, RoPE-based Transformers with poly(n)-precision, $O(1)$ layers, and hidden dimension $d \leq O(n)$ cannot solve either arithmetic formula evaluation or Boolean formula value problems. This result is particularly significant as it reveals fundamental limitations in the expressivity of RoPE-based architectures, despite their empirical success in modern language models.

7 Discussion

One limitation is that our analysis focuses primarily on the forward computation aspects, which and assumes constant-depth nonlinear activation functions, leaving open questions about training dynamics and the impact of more complex activation functions. It would be interesting to extend our theoretical results to analyze other variants of positional embeddings and investigate whether similar complexity bounds hold for more sophisticated Transformer architectures. Furthermore, our results suggest a potential gap between theoretical limitations and empirical performance.

References

[AAA+23] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[AB09] Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

[Ant24] Anthropic. The claude 3 model family: Opus, sonnet, haiku, 2024. https://www-cdn.anthropic.com/de8ba9b01c9ab7cbabf5c33b80b7bbc618857627/Model_Card_C

[AS23a] Josh Alman and Zhao Song. Fast attention requires bounded entries. In *NeurIPS*, 2023.

[AS23b] Josh Alman and Zhao Song. How to capture higher-order correlations? generalizing matrix softmax attention to kronecker computation. *arXiv preprint arXiv:2310.04064*, 2023.

[AS24a] Josh Alman and Zhao Song. Fast rope attention: Combining the polynomial method and fast fourier transform. *manuscript*, 2024.

[AS24b] Josh Alman and Zhao Song. The fine-grained complexity of gradient computation for training large language models. In *NeurIPS*, 2024.

[BAY21] Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks? *arXiv preprint arXiv:2105.14491*, 2021.

[BBH+22] Sid Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, et al. Gpt-neox-20b: An open-source autoregressive language model. *arXiv preprint arXiv:2204.06745*, 2022.

[BCGR92] S Buss, S Cook, Arvind Gupta, and Vijaya Ramachandran. An optimal parallel algorithm for formula evaluation. *SIAM Journal on Computing*, 21(4):755–780, 1992.

[BI94] D Mix Barrington and Neil Immerman. Time, hardware, and uniformity. In *Proceedings of IEEE 9th Annual Conference on Structure in Complexity Theory*, pages 176–185. IEEE, 1994.

[BSA+23] Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pages 2397–2430. PMLR, 2023.

[Bus87] Samuel R Buss. The boolean formula value problem is in alogtime. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pages 123–131, 1987.

[CAM24] Anshuman Chhabra, Hadi Askari, and Prasant Mohapatra. Revisiting zero-shot abstractive summarization in the era of large language models from the perspective of position bias. *arXiv preprint arXiv:2401.01989*, 2024.

[Cha22] François Charton. What is my math transformer doing?–three results on interpretability and generalization. *arXiv preprint arXiv:2211.00170*, 2022.

[Chi24] David Chiang. Transformers in dlogtime-uniform tc0. *arXiv preprint arXiv:2409.13629*, 2024.

[CHL+24] Ya-Ting Chang, Zhibo Hu, Xiaoyu Li, Shuiqiao Yang, Jiaojiao Jiang, and Nan Sun. Dihan: A novel dynamic hierarchical graph attention network for fake news detection. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pages 197–206, 2024.

[CLL+24] Bo Chen, Xiaoyu Li, Yingyu Liang, Zhenmei Shi, and Zhao Song. Bypassing the exponential dependency: Looped transformers efficiently learn in-context by multi-step gradient descent. *arXiv preprint arXiv:2410.11268*, 2024.

[CLS+24] Bo Chen, Yingyu Liang, Zhizhou Sha, Zhenmei Shi, and Zhao Song. Hsr-enhanced sparse attention acceleration. *arXiv preprint arXiv:2410.10165*, 2024.

[CND+23] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113, 2023.

[CYL+24] Weize Chen, Ziming You, Ran Li, Yitong Guan, Chen Qian, Chenyang Zhao, Cheng Yang, Ruobing Xie, Zhiyuan Liu, and Maosong Sun. Internet of agents: Weaving a web of heterogeneous agents for collaborative intelligence. *arXiv preprint arXiv:2407.07061*, 2024.

[DSWY22] Yichuan Deng, Zhao Song, Yitan Wang, and Yuanyuan Yang. A nearly optimal size coreset algorithm with nearly linear time. *arXiv preprint arXiv:2210.08361*, 2022.

[FJL+24] Tao Feng, Chuanyang Jin, Jingyu Liu, Kunlun Zhu, Haoqin Tu, Zirui Cheng, Guanyu Lin, and Jiaxuan You. How far are we from agi. *arXiv preprint arXiv:2405.10313*, 2024.

[FZG+23] Guhao Feng, Bohang Zhang, Yuntian Gu, Haotian Ye, Di He, and Liwei Wang. Towards revealing the mystery behind chain of thought: a theoretical perspective. *Advances in Neural Information Processing Systems*, 36, 2023.

[GMS23] Yeqi Gao, Sridhar Mahadevan, and Zhao Song. An over-parameterized exponential regression. *arXiv preprint arXiv:2303.16504*, 2023.

[GXG+23] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2023.

[HAB02] William Hesse, Eric Allender, and David A Mix Barrington. Uniform constant-depth threshold circuits for division and iterated multiplication. *Journal of Computer and System Sciences*, 65(4):695–716, 2002.

[HCL+24] Jerry Yao-Chieh Hu, Pei-Hsuan Chang, Haozheng Luo, Hong-Yu Chen, Weijian Li, Wei-Po Wang, and Han Liu. Outlier-efficient hopfield layers for large transformer-based models. In *Forty-first International Conference on Machine Learning (ICML)*, 2024.

[HCW+24] Jerry Yao-Chieh Hu, Bo-Yu Chen, Dennis Wu, Feng Ruan, and Han Liu. Nonparametric modern hopfield models. *arXiv preprint arXiv:2404.03900*, 2024.

[HLSL24] Jerry Yao-Chieh Hu, Thomas Lin, Zhao Song, and Han Liu. On computational limits of modern hopfield models: A fine-grained complexity analysis. In *Forty-first International Conference on Machine Learning (ICML)*, 2024.

[HSK+24] Jerry Yao-Chieh Hu, Maojiang Su, En-Jui Kuo, Zhao Song, and Han Liu. Computational limits of low-rank adaptation (lora) for transformer-based models. *arXiv preprint arXiv:2406.03136*, 2024.

[HWL24a] Jerry Yao-Chieh Hu, Dennis Wu, and Han Liu. Provably optimal memory capacity for modern hopfield models: Transformer-compatible dense associative memories as spherical codes. In *Thirty-eighth Conference on Neural Information Processing Systems (NeurIPS)*, 2024.

[HWL+24b] Jerry Yao-Chieh Hu, Weimin Wu, Zhuoru Li, Sophia Pi, , Zhao Song, and Han Liu. On statistical rates and provably efficient criteria of latent diffusion transformers (dits). In *Thirty-eighth Conference on Neural Information Processing Systems (NeurIPS)*, 2024.

[HYW+23] Jerry Yao-Chieh Hu, Donglin Yang, Dennis Wu, Chenwei Xu, Bo-Yu Chen, and Han Liu. On sparse modern hopfield model. In *Thirty-seventh Conference on Neural Information Processing Systems (NeurIPS)*, 2023.

[Imm98] Neil Immerman. *Descriptive complexity*. Springer Science & Business Media, 1998.

[IP01] Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-sat. *Journal of Computer and System Sciences*, 62(2):367–375, 2001.

[KGR⁺22] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213, 2022.

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

[LAG⁺22] Bingbin Liu, Jordan T Ash, Surbhi Goel, Akshay Krishnamurthy, and Cyril Zhang. Transformers learn shortcuts to automata. *arXiv preprint arXiv:2210.10749*, 2022.

[LCT⁺24] Na Liu, Liangyu Chen, Xiaoyu Tian, Wei Zou, Kaijiang Chen, and Ming Cui. From llm to conversational agent: A memory enhanced architecture with fine-tuning of large language models. *arXiv preprint arXiv:2401.02777*, 2024.

[LLS⁺24a] Chenyang Li, Yingyu Liang, Zhenmei Shi, Zhao Song, and Tianyi Zhou. Fourier circuits in neural networks and transformers: A case study of modular arithmetic with multiple inputs. *arXiv preprint arXiv:2402.09469*, 2024.

[LLS⁺24b] Xiaoyu Li, Yingyu Liang, Zhenmei Shi, Zhao Song, and Junwei Yu. Fast john ellipsoid computation with differential privacy optimization. *arXiv preprint arXiv:2408.06395*, 2024.

[LLS⁺24c] Xiaoyu Li, Yingyu Liang, Zhenmei Shi, Zhao Song, and Yufa Zhou. Fine-grained attention i/o complexity: Comprehensive analysis for backward passes. *arXiv preprint arXiv:2410.09397*, 2024.

[LLS⁺24d] Yingyu Liang, Heshan Liu, Zhenmei Shi, Zhao Song, Zhuoyan Xu, and Junze Yin. Conv-basis: A new paradigm for efficient attention inference and gradient computation in transformers. *arXiv preprint arXiv:2405.05219*, 2024.

[LLS⁺24e] Yingyu Liang, Jiangxuan Long, Zhenmei Shi, Zhao Song, and Yufa Zhou. Beyond linear approximations: A novel pruning approach for attention matrix. *arXiv preprint arXiv:2410.11261*, 2024.

[LLSS24] Xiaoyu Li, Yingyu Liang, Zhenmei Shi, and Zhao Song. A tighter complexity analysis of sparsegpt. *arXiv preprint arXiv:2408.12151*, 2024.

[LLZM24] Zhiyuan Li, Hong Liu, Denny Zhou, and Tengyu Ma. Chain of thought empowers transformers to solve inherently serial problems. In *The Twelfth International Conference on Learning Representations*, 2024.

[LPP⁺20] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474, 2020.

[LSS+24a] Yingyu Liang, Zhizhou Sha, Zhenmei Shi, Zhao Song, and Yufa Zhou. Looped relu mlps may be all you need as practical programmable computers. *arXiv preprint arXiv:2410.09375*, 2024.

[LSS+24b] Yingyu Liang, Zhizhou Sha, Zhenmei Shi, Zhao Song, and Yufa Zhou. Looped relu mlps may be all you need as practical programmable computers. *arXiv preprint arXiv:2410.09375*, 2024.

[LSS+24c] Yingyu Liang, Zhizhou Sha, Zhenmei Shi, Zhao Song, and Yufa Zhou. Multi-layer transformers gradient can be approximated in almost linear time. *arXiv preprint arXiv:2408.13233*, 2024.

[LSSY24] Yingyu Liang, Zhenmei Shi, Zhao Song, and Chiwun Yang. Toward infinite-long prefix in transformer. *arXiv preprint arXiv:2406.14036*, 2024.

[LSSZ24a] Yingyu Liang, Zhenmei Shi, Zhao Song, and Yufa Zhou. Differential privacy of cross-attention with provable guarantee. *arXiv preprint arXiv:2407.14717*, 2024.

[LSSZ24b] Yingyu Liang, Zhenmei Shi, Zhao Song, and Yufa Zhou. Tensor attention training: Provably efficient learning of higher-order transformers. *arXiv preprint arXiv:2405.16411*, 2024.

[LT24] AI @ Meta Llama Team. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[MLT+24] Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating very long-term conversational memory of llm agents. *arXiv preprint arXiv:2402.17753*, 2024.

[MS23a] William Merrill and Ashish Sabharwal. A logic for expressing log-precision transformers. *Advances in Neural Information Processing Systems*, 36, 2023.

[MS23b] William Merrill and Ashish Sabharwal. The parallelism tradeoff: Limitations of log-precision transformers. *Transactions of the Association for Computational Linguistics*, 11:531–545, 2023.

[MSS22] William Merrill, Ashish Sabharwal, and Noah A Smith. Saturated transformers are constant-depth threshold circuits. *Transactions of the Association for Computational Linguistics*, 10:843–856, 2022.

[Ope24] OpenAI. Introducing openai o1-preview. https://openai.com/index/introducing-openai-o1-pre 2024. Accessed: September 12.

[SAL+24] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

[Sip96] Michael Sipser. Introduction to the theory of computation. *ACM Sigact News*, 27(1):27–29, 1996.

[SMN+24] Zhenmei Shi, Yifei Ming, Xuan-Phi Nguyen, Yingyu Liang, and Shafiq Joty. Discovering the gems in early layers: Accelerating long-context llms with 1000x input token reduction. *arXiv preprint arXiv:2409.17422*, 2024.

[SSX23] Anshumali Shrivastava, Zhao Song, and Zhaozhuo Xu. A theoretical analysis of nearest neighbor search on approximate near neighbor graph. *arXiv preprint arXiv:2303.06210*, 2023.

[SY23] Zhao Song and Chiwun Yang. An automatic learning rate schedule algorithm for achieving faster convergence and steeper descent. *arXiv preprint arXiv:2310.11291*, 2023.

[SZZ24] Zhao Song, Lichen Zhang, and Ruizhe Zhang. Training multi-layer over-parametrized neural network in subquadratic time. In *Innovations in Theoretical Computer Science (ITCS)*, pages 93:1–93:15, 2024.

[VCC+17] Petar Velickovi, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[Vol99] Heribert Vollmer. *Introduction to circuit complexity: a uniform approach.* Springer Science & Business Media, 1999.

[VSP+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, volume 30, 2017.

[WHHL24] Dennis Wu, Jerry Yao-Chieh Hu, Teng-Yun Hsiao, and Han Liu. Uniform memory retrieval with larger capacity for modern hopfield models. In *Forty-first International Conference on Machine Learning (ICML)*, 2024.

[WHL+24] Dennis Wu, Jerry Yao-Chieh Hu, Weijian Li, Bo-Yu Chen, and Han Liu. STanhop: Sparse tandem hopfield model for memory-enhanced time series prediction. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024.

[Wil18] Virginia Vassilevska Williams. On some fine-grained questions in algorithms and complexity. In *Proceedings of the international congress of mathematicians: Rio de janeiro 2018*, pages 3447–3487. World Scientific, 2018.

[WJS+19] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. Heterogeneous graph attention network. In *The world wide web conference*, pages 2022–2032, 2019.

[WWS+22] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[XCG+23] Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, et al. The rise and potential of large language model based agents: A survey. *arXiv preprint arXiv:2309.07864*, 2023.

[XGW+22] Xinchao Xu, Zhibin Gou, Wenquan Wu, Zheng-Yu Niu, Hua Wu, Haifeng Wang, and Shihang Wang. Long time no see! open-domain conversation with long-term persona memory. *arXiv preprint arXiv:2203.05797*, 2022.

[XHH+24] Chenwei Xu, Yu-Chao Huang, Jerry Yao-Chieh Hu, Weijian Li, Ammar Gilani, Hsi-Sheng Goan, and Han Liu. Bishop: Bi-directional cellular learning for tabular data with generalized sparse modern hopfield model. In *Forty-first International Conference on Machine Learning (ICML)*, 2024.

[XSL24] Zhuoyan Xu, Zhenmei Shi, and Yingyu Liang. Do large language models have compositional ability? an investigation into limitations and scalability. In *First Conference on Language Modeling*, 2024.

[ZJV+24] Chenyang Zhao, Xueying Jia, Vijay Viswanathan, Tongshuang Wu, and Graham Neubig. Self-guide: Better task-specific instruction following via self-synthetic finetuning. *arXiv preprint arXiv:2407.12874*, 2024.

Appendix

A Proof of Lemma 4.1

Here we present the proof of Lemma 4.1. We restate Lemma 4.1 below

Lemma A.1 (Trigonometric function approximation in TC^0, formal version of Lemma 4.1). *If $p \leq \mathrm{poly}(n)$, then for every p-bit floating point number x, there is a constant-depth uniform threshold circuit of size $\mathrm{poly}(n)$ which can compute $\sin(x)$ and $\cos(x)$ with a relative error at most 2^{-p}. We use $d_\triangle$ denote the maximum depth needed for computing $\sin(x)$ and $\cos(x)$.*

Proof. For $\sin(x)$ where $x \in \mathbb{F}_p$, we can define: $k := \left\lfloor \frac{x}{2/\pi} \right\rfloor$ and

$$
r := \begin{cases} x - k\pi/2 & \text{if } x - k\pi/2 \leq \pi/4, \\ (k+1)\pi/2 - x & \text{else.} \end{cases}
$$

Using truncated Taylor series of $\sin(r)$, we have:

$$
\sin(r) = \sum_{i=0}^{N-1} (-1)^i \frac{r^{2i+1}}{(2i+1)!} + R_N^{\sin}(r)
$$

For $R_N^{\sin}(r)$, we can show:

$$
\begin{aligned}
R_N^{\sin}(r) &\leq (\pi/4)^{2N+1} \frac{1}{(2N+1)!} \\
&\leq \frac{1}{(2N+1)!} \\
&= O(1/N!) \\
&\leq O(2^{-N})
\end{aligned}
$$

where the first step follows from the definition of the Lagrange remainder term, the second step follows from $(\pi/4)^{2N+1} \leq 1$, the fourth step follows from $O(2^x) < O(x!)$ holds for any positive x.

Similarly, using truncated Taylor series of $\cos(r)$, we have:

$$
\cos(r) = \sum_{i=0}^{N-1} (-1)^i \frac{r^{2i}}{(2i)!} + R_N^{\cos}(r)
$$

For $R_N^{\cos}(r)$, we can show:

$$
\begin{aligned}
R_N^{\cos}(r) &\leq (\pi/4)^{2N} \frac{1}{(2N)!} \\
&\leq \frac{1}{(2N)!} \\
&= O(1/N!) \\
&\leq O(2^{-N})
\end{aligned}
$$

where the first step follows from the definition of the Lagrange remainder term, the second step follows from $(\pi/4)^{2N+1} \leq 1$, the fourth step follows from $O(2^x) < O(x!)$ holds for any positive x. Then, we have

$$\sin(x) = \begin{cases} \sin(r) & \text{if } x - k\pi/2 \leq \pi/4, \\ \cos(r) & \text{else.} \end{cases}$$

and

$$\cos(x) = \begin{cases} \cos(r) & \text{if } x - k\pi/2 \leq \pi/4, \\ \sin(r) & \text{else.} \end{cases}$$

Because of similar calculation step between $\sin(x)$ or $\cos(x)$, we can show the depth of circuit to compute them following from Lemma 3.16 and Corollary 3.17:

1. To get the value of k, we need to calculate floor and division, which use depth-$2d_{\text{std}}$ circuit.

2. To get the value of r, we need to calculate addition, comparison, multiplication and division, which use depth-$4d_{\text{std}}$ circuit.

3. To get the value of $\sin(r)$ or $\cos(r)$, we need to calculate addition and iterated addition. For each entry in iterated addition, we need to calculate multiplication, division and iterated multiplication in parallel, which use depth-$(3d_{\text{std}} + d_\otimes + d_\oplus)$ circuit.

4. To get the value of $\sin(x)$ or $\cos(x)$, we need to calculate comparison, which use depth-d_{std} circuit.

Finally, we can show

$$d_\triangle = 8d_{\text{std}} + d_\oplus + d_\otimes.$$

Thus we complete the proof. □

B Mssing proofs in Section 4

Here we present some missing proofs in Section 4. First we show the proof of Lemma 4.2 below.

Lemma B.1 (Matrix multiplication in TC^0, formal version of Lemma 4.2). *Let $A \in \mathbb{F}_p^{n_1 \times d}$, $B \in \mathbb{F}_p^{d \times n_2}$ be two matrices. If $p \leq \text{poly}(n), n_1, n_2 \leq \text{poly}(n), d \leq n$, then AB can be computable by a uniform threshold circuit with size $\text{poly}(n)$ and depth $(d_{\text{std}} + d_\oplus)$.*

Proof. For each $i \in [n_1]$ and $j \in [n_2]$, the entry $(AB)_{i,j}$ is given by $(AB)_{i,j} = \sum_{k=1}^{d} A_{i,k} B_{k,j}$. By Part 1 of Lemma 3.16, each product $A_{i,k} B_{k,j}$ can be computed by a uniform threshold circuit of depth d_{std}. Since these products for different k can be computed in parallel, all products $A_{i,k} B_{k,j}$ for $k \in [d]$ can be computed simultaneously in depth d_{std}.

Next, by Part 3 of Lemma 3.16, the sum $\sum_{k=1}^{d} A_{i,k} B_{k,j}$ can be computed by a uniform threshold circuit of depth $d_\oplus$. Therefore, the total depth required to compute $(AB)_{i,j}$ is $d_{\text{std}} + d_\oplus$. Since we can compute $(AB)_{i,j}$ for all $i \in [n_1]$ and $j \in [n_2]$ in parallel, the overall depth of the circuit remains $d_{\text{std}} + d_\oplus$. The size of the circuit is polynomial in n because $n_1, n_2, d \leq \text{poly}(n)$, and each operation is computed by a circuit of polynomial size. Therefore, AB can be computed by a uniform threshold circuit with size $\text{poly}(n)$ and depth $d_{\text{std}} + d_\oplus$. Thus we complete the proof. □

Here we state the proof of Lemma 4.3.

Lemma B.2 (RoPE-based attention matrix computation in TC^0, formal version of Lemma 4.3). *If $p \leq \mathrm{poly}(n)$, then the attention matrix A in Definition 3.23 can be computable by a uniform threshold circuit with size $\mathrm{poly}(n)$ and depth $4(d_{\mathrm{std}} + d_{\oplus}) + d_{\triangle} + d_{\exp}$.*

Proof. For each $i, j \in [n]$, we need to compute the entry $A_{i,j}$ of the attention matrix A as defined in Definition 3.23.

By Lemma 4.1, each entry of R_{j-i} can be computed using a uniform threshold circuit of size $\mathrm{poly}(n)$ and depth $d_{\triangle}$. Since $n \leq \mathrm{poly}(n)$, all entries of R_{j-i} can be computed in parallel with the same circuit size and depth.

Using Lemma 4.2, the matrix product $W_Q R_{j-i}$ can be computed by a uniform threshold circuit of size $\mathrm{poly}(n)$ and depth $d_{\mathrm{std}} + d_{\oplus}$.

Applying Lemma 4.2 again, the product $(W_Q R_{j-i}) W_K^{\top}$ can be computed with the same circuit size and depth $d_{\mathrm{std}} + d_{\oplus}$.

Next, the scalar product

$$s_{i,j} = X_{i,*}(W_Q R_{j-i} W_K^{\top}) X_{j,*}^{\top}$$

can be computed using a uniform threshold circuit of size $\mathrm{poly}(n)$ and depth $2(d_{\mathrm{std}} + d_{\oplus})$, again by Lemma 4.2.

Using Lemma 3.18, the exponential function $A_{i,j} = \exp(s_{i,j})$ can be computed by a uniform threshold circuit of size $\mathrm{poly}(n)$ and depth $d_{\exp}$.

Combining the depths from each step, the total depth required to compute $A_{i,j}$ is

$$d_{\mathrm{total}} = 4(d_{\mathrm{std}} + d_{\oplus}) + d_{\triangle} + d_{\exp}.$$

Since all entries $A_{i,j}$ for $i, j \in [n]$ can be computed in parallel, the overall circuit has size $\mathrm{poly}(n)$ and depth $4(d_{\mathrm{std}} + d_{\oplus}) + d_{\triangle} + d_{\exp}$. Therefore, the attention matrix A can be computed by a uniform threshold circuit with size $\mathrm{poly}(n)$ and depth $4(d_{\mathrm{std}} + d_{\oplus}) + d_{\triangle} + d_{\exp}$.

Thus we complete the proof. $\square$

Here we state the proof of Lemma 4.5.

Lemma B.3 (MLP computation in TC^0, formal version of Lemma 4.5). *If $p \leq \mathrm{poly}(n)$, then the MLP layer in Definition 3.26 can be computable by a uniform threshold circuit with size $\mathrm{poly}(n)$ and depth $2d_{\mathrm{std}} + d_{\oplus}$.*

Proof. For each $i \in [m]$, by Lemma 4.2, we need a circuit with depth $d_{\mathrm{std}} + d_{\oplus}$ and size $\mathrm{poly}(n)$ to compute $W X_{i,*}$, and by Part 1 of Lemma 3.16, w need a circuit with depth d_{std} and size $\mathrm{poly}(n)$ to compute $W X_{i,*} + b$. Hence the total depth need is $2d_{\mathrm{std}} + d_{\oplus}$ and total size is still $\mathrm{poly}(n)$. Since this procedure can be done in parallel for all $i \in [n]$, the proof is complete. $\square$

Here we state the proof of Lemma 4.6.

Lemma B.4 (Layer-norm computation in TC^0, formal version of Lemma 4.6). *If $p \leq \mathrm{poly}(n)$, then the Layer-wise Normalization layer in Definition 3.27 can be computable by a uniform threshold circuit with size $\mathrm{poly}(n)$ and depth $5d_{\mathrm{std}} + 2d_{\oplus} + d_{\mathrm{sqrt}}$.*

Proof. For each $i \in [n]$, by Lemma 3.16, we can compute μ_i using a circuit with depth $d_{\mathrm{std}} + d_{\oplus}$ and size $\mathrm{poly}(n)$ and then compute σ_i^2 with depth $2d_{\mathrm{std}} + d_{\oplus}$ and size $\mathrm{poly}(n)$. By Lemma 3.16 and Lemma 3.19, we can compute $g^{\mathrm{LN}}(x)_{i,*}$ using a circuit with depth $2d_{\mathrm{std}} + d_{\mathrm{sqrt}}$ and size $\mathrm{poly}(n)$. Hence the total needed depth is $5d_{\mathrm{std}} + 2d_{\oplus} + d_{\mathrm{sqrt}}$ and size is $\mathrm{poly}(n)$. Since this procedure can be done in parallel for all $i \in [n]$, the proof is complete. $\square$

Here we state the proof of Lemma 4.7.

Lemma B.5 (Multi-layer RoPE-based Transformer computation in TC^0, formal version of Lemma 4.7). *Suppose that for each $i \in [m]$, g_i in TF is computable by a constant depth d_g uniform threshold circuit with size $\mathrm{poly}(n)$. If $p \leq \mathrm{poly}(n)$, then the RoPE-based Transformer TF in Definition 3.25 can be computable by a uniform threshold circuit with size $\mathrm{poly}(n)$ and depth $(m+1)d_g + 7m(d_{\mathrm{std}} + d_{\oplus}) + m(d_{\triangle} + d_{\exp})$.*

Proof. For each $i \in [m]$, by condition, g_i is computable by a constant depth d_g uniform threshold circuit with size $\mathrm{poly}(n)$.

For each $i \in [m]$, by Lemma 4.4, Attn_i is computable by a uniform threshold circuit with depth $7(d_{\mathrm{std}} + d_{\oplus}) + d_{\triangle} + d_{\exp}$ and size $\mathrm{poly}(n)$.

Hence, to compute $\mathsf{TF}(X)$, we need to compute $g_0, g_1, \ldots, g_m$ and $\mathsf{Attn}_1, \ldots, \mathsf{Attn}_m$, thus the total depth of the circuit is $(m+1)d_g + 7m(d_{\mathrm{std}} + d_{\oplus}) + m(d_{\triangle} + d_{\exp})$ and the size of circuit is $\mathrm{poly}(n)$.

Thus we complete the proof. $\qquad\square$