Circuit Complexity Bounds for RoPE-based Transformer Architecture

Bo Chen **Xiaoyu Li** **Yingyu Liang*** **Jiangxuan Long**

Zhenmei Shi **Zhao Song†** **Jiahao Zhang**

Abstract

Characterizing the expressive 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, position embedding has emerged as a crucial technique in modern large language models, offering superior performance in capturing positional information, which shows great performance for the long context scenario. In this work, we take a circuit complexity perspective and rigorously analyze Transformers augmented with widely adopted positional embeddings. We prove that, under standard complexity assumptions, such models remain incapable of efficiently solving canonical tasks such as arithmetic formula evaluation and Boolean formula value computation. Our results expose a fundamental expressivity limitation that persists despite the remarkable empirical success of positionally-enhanced Transformers. Beyond tightening known complexity bounds, our findings offer new theoretical insights for designing future architectures with provably stronger reasoning and compositional capabilities.

1 Introduction

Recently, Large Language Models (LLMs), such as GPT-4 (Achiam et al., 2023), Claude (Anthropic, 2024), Llama (Llama Team, 2024), and more recently, OpenAI's o1 (OpenAI, 2024b), have exhibited remarkable potential to revolutionize numerous facets of daily life, including conversational AI (Liu et al., 2024), AI agents (Xi et al., 2023; Chen et al., 2024), search capabilities (OpenAI, 2024b), and AI assistants (Kuo et al., 2024; Feng et al., 2024), 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 (Chhabra et al., 2024; Zhao et al., 2024) and sustaining coherent, extended conversations (Xu et al., 2022; Maharana et al., 2024). The o1 model from OpenAI (OpenAI, 2024b) represents a major advancement in this field. By leveraging Chain-of-Thought (CoT) reasoning (Wei et al., 2022; Kojima et al., 2022) and incorporating Retrieval Augmented Generation (RAG) (Lewis et al., 2020; Gao et al., 2023), it showcases a level of expertise comparable to PhD-level problem solving, with both techniques heavily relying on extensive contextual understanding.

LLMs are primarily built upon the Transformer architecture (Vaswani et al., 2017), 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? To address the aforementioned questions and to investigate the expressiveness of transformers, prior research has made significant strides. For example, (Merrill and Sabharwal, 2023b) 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, (Merrill and Sabharwal, 2023a) demonstrate that $\mathsf{DLOGTIME}$-uniform TC^0 circuits are capable of simulating softmax-attention transformers. Building on this foundation, (Chiang, 2024) refine these results by in-

*Corresponding author: yingyul@hku.hk. HKU. yliang@cs.wisc.edu. UW-Madison.

†Corresponding author: magic.linuxkde@gmail.com. University of California, Berkeley.

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 11080–11097
November 4-9, 2025

creasing 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.

Position embeddings (Vaswani et al., 2017) are a critical component in determining the representational power of Transformers, as they provide the model with information about token order. Recent works such as (Biderman et al., 2023; Hua et al., 2025) can lead to substantially powerful capabilities in modeling long-range interactions and generalizing to longer sequences. Despite the considerable empirical success of positional embeddings, it is still unclear whether they increase a model's fundamental expressive power; thus, a natural question arises:

How do position embeddings affect the expressiveness of Large Language Models?

To make this question concrete, we formalize a widely adopted positional encoding, Rotary Position Embedding (RoPE) (Su et al., 2024), as our canonical case. By encoding positional information via rotation matrices in the query-key space, RoPE captures both absolute and relative positions, thereby improving inductive bias for attention, enabling better length generalization, and enhancing performance on long-context tasks. This work aims to address the proposed question from the perspective of circuit complexity on RoPE, taking a step toward a principled understanding of the computational power of position embeddings of Transformers. We present a rigorous theoretical analysis that establishes fundamental limits on their expressiveness and clarifies the role of positional encoding in shaping the model's capabilities.

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, 2023b,a) and (Chiang, 2024), our contribution are summarized as follows:

- We prove that under standard complexity assumptions, 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 under standard complexity assumptions, 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 5.1).
- We prove that under standard complexity assumptions, 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 5.2).

2 Related Work

2.1 Complexity and Neural Networks

Circuit complexity, a branch of computational complexity theory, studies circuit families as models of computation (Li et al., 2025; Ke et al., 2025; Chen et al., 2025c). 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 (Merrill et al., 2022). 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, (Liu et al., 2022) shows that Transformer depth must depend on input sequence length when simulating non-solvable semiautomata. (Li et al., 2024) 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 (Impagliazzo and Paturi, 2001), 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 (Williams, 2018) and has been applied to derive lower bounds for other problems (Deng et al., 2025; Liang et al., 2025; Chen et al., 2025a,b). Specifically, (Alman and Song, 2023) 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, (Alman and Song, 2024) 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 neural networks' computational capabilities by rigorously analyzing the computational problems they can efficiently solve.

2.2 Limitations of Transformers

Transformers have shown exceptional capabilities in natural language processing tasks, yet their effectiveness in mathematical computations remains limited (Charton, 2022). 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 (Merrill et al., 2022) 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, (Liu et al., 2022) prove that softmax-attention Transformers can be simulated by a non-uniform TC^0 circuit. Extending this analysis, (Merrill and Sabharwal, 2023b) 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 (Immerman, 1998)), (Merrill and Sabharwal, 2023a) demonstrate that DLOGTIME-uniform TC^0 can simulate softmax-attention Transformers. (Chiang, 2024) 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 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, (Feng et al., 2023) 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 (Sipser, 1996). These theoretical constraints help explain some of the practical limitations observed when applying Transformers to mathematical tasks.

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. 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 $X \in \mathbb{R}^{m \times n}$. 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

In this section, we formally define key concepts in circuit complexity. For enhanced readability, Appendix A also provides an intuitive, self-contained overview of this field.

The Boolean circuit is formally defined as:

Definition 3.1 (Boolean circuit, Definition 6.1 on page 102 of (Arora and Barak, 2009))**.** *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 (Arora and Barak, 2009))**.** *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 (Arora and Barak, 2009))**.** *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 (Arora and Barak, 2009))**.** *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 (Vollmer, 1999))**.** *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 with prime values. When a Boolean circuit is equipped with any of them, we call it a threshold circuit.*

Finally, we recall the definition of P class.

Definition 3.7 (P, Definition 1.20 on page 9 of (Arora and Barak, 2009))**.** *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 (Arora and Barak, 2009), Corollary 4.35 on page 126 of (Vollmer, 1999))**.** *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 (Arora and Barak, 2009), page 116 in (Vollmer, 1999) 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 (Arora and Barak, 2009))**.** *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 (Barrington and Immerman, 1994))**.** *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 (Barrington and Immerman, 1994; Hesse et al., 2002) 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 (Chiang, 2024))**.** *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* $\mathbb{F}_p$*.*

These floating-point numbers are not just theoretical constructs, and they can be efficiently implemented in hardware. For more details on their basic operations (e.g., rounding, arithmetic operations), please refer to Appendix B.

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 3.14 (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 3.15 (Rotation matrix block)**.** *Let* $\theta \in \mathbb{F}_p$*. For a length-*n *input sequence with embedding dimension* d*, 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 3.16 (Rotation matrix)**.** *Let* j *be the index of position in the sequence,* i *the index of tokens, we define the overall relative rotation matrix*

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

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 (Su et al., 2024).*

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

Definition 3.17 (RoPE attention matrix)**.** *Let Rotation matrix* R_{j-i} *be defined in Definition 3.16. 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 3.18 (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*

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

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

Definition 3.19 (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 3.18). 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*

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

where $\circ$ denotes function composition.

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

Definition 3.20 (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 3.21 (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 Lemma E.1 in Appendix E)**.** *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 E.2 in Appendix E)**.** *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 E.3 in Appendix E)**.** *If $p \leq \mathrm{poly}(n)$, then the attention matrix A in Definition 3.17 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, informal version of Lemma E.4 in Appendix E)**.** *If $p \leq \mathrm{poly}(n)$, then the attention layer Attn in Definition 3.18 can be computable by a uniform threshold circuit with size $\mathrm{poly}(n)$ and depth $7(d_{\mathrm{std}} + d_{\oplus}) + d_{\triangle} + d_{\exp}$.*

4.5 Computing Common Buliding Blocks other than Self-attention layer

In Definition 3.19, 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 E.5 in Appendix E)**.** *If $p \leq \mathrm{poly}(n)$, then the MLP layer in Definition 3.20 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 E.6 in Appendix E)**.** *If $p \leq \mathrm{poly}(n)$, then the Layer-wise Normalization layer in Definition 3.21 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 E.7 in Appendix E)**.** *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 3.19 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, informal version of Theorem E.8 in Appendix E)**.** *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 3.19 can be simulated by a uniform TC^0 circuit family.*

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 $\mathsf{DLOGTIME}$-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 present our two main hardness results, focusing on two key problems: the Arithmetic Formula Evaluation problem and the Boolean Formula Evaluation problem. For detailed definitions, refer to Appendix C and Appendix D.

Theorem 5.1. *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 C.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 5.2. *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 D.4 (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. □

The above two theorems show the computation limitation of RoPE-based Transformer unless $\mathsf{TC}^0 = \mathsf{NC}^1$. Beyond RoPE, the analysis applies to constant-time, parallelizable positional embeddings; we outline the adaptation and illustrate it for ALiBi in Appendix F (Press et al., 2022).

6 Discussion

In this section, we discuss the relevance and practical implications of our work.

Relevance to the NLP Community. Circuit complexity has recently gained attention in theoretical NLP research, with several works published at major NLP venues (Merrill et al., 2021; Merrill and Sabharwal, 2023b; Hao et al., 2022). These studies demonstrate how circuit complexity provides insights into both the capabilities and limitations of modern LLMs. Our work extends this line of research to positional encoding mechanisms, an essential but relatively under-theorized component of contemporary language models.

Practical Implications of Hardness Results. We want to first establish that NC^1 is tightly associated with the reasoning problem. Notably, many tasks in widely-used mathematical reasoning benchmarks such as MathQA (Amini et al., 2019) and NumGLUE (Mishra et al., 2022) fundamentally involve arithmetic formula evaluation problems–tasks which inherently belong to NC^1.

Our work aims to bridge theoretical insights from circuit complexity with practical limitations in LLMs, especially those using RoPE. While RoPE is widely adopted in cutting-edge models such as Qwen (Bai et al., 2023), Llama (Meta, 2024), and ChatGPT (OpenAI, 2024a), its theoretical limitations remain largely unexplored. By proving that RoPE-based Transformers are unlikely to solve NC^1-complete problems with constant depth, we reveal an inherent computational bottleneck that is independent of parameter count or training data.

Insights for Model Design. Our main results (Theorem 5.1 and Theorem 5.2) and additional results in Appendix F establish that Transformers with advanced positional embeddings remain in TC^0 and therefore cannot solve a broad class of reasoning problems, including fundamental arithmetic computations. This contradicts prior claims that richer positional encodings improve reasoning ability (Ke et al., 2021; Zhu et al., 2025), and suggests that explicit positional embeddings may not be necessary for core reasoning functions.

Our findings are consistent with recent work demonstrating that Transformers without any positional embeddings can still capture positional information implicitly (Haviv et al., 2022) and can even exhibit superior extrapolation on algorithmic and mathematical reasoning benchmarks (Kazemnejad et al., 2023). Moving forward, we recommend that empirical research prioritize chain-of-thought methods for enhancing reasoning performance (Li et al., 2024) rather than exploring a wide variety of positional encoding schemes.

7 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 $\mathrm{poly}(n)$-precision, $O(1)$ layers, and hidden dimension $d \leq O(n)$ cannot solve either arithmetic formula evaluation or Boolean formula value problems. This important theoretical result goes beyond the empirical success of RoPE-based architectures, revealing fundamental limitations in their expressivity and providing insights for future model design.

Limitation

One limitation is that our analysis focuses primarily on the forward computation of constant-depth standard Transformers, leaving interesting open questions about extending our framework to other variants of Transformer architectures.

Potential Risks

In this paper, we discuss the circuit complexity of RoPE-based Transformer architectures and present several theoretical hardness results showing that they cannot solve certain practical reasoning problems. Since this paper is entirely theoretical, we do not foresee any negative societal impacts.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Josh Alman and Zhao Song. 2023. Fast attention requires bounded entries. In *NeurIPS*.

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

Aida Amini, Saadia Gabriel, Shanchuan Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. 2019. Mathqa: Towards interpretable math word problem solving with operation-based formalisms. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2357–2367.

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

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

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, and 1 others. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

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

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

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

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

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

Bo Chen, Xiaoyu Li, Yingyu Liang, Zhenmei Shi, and Zhao Song. 2025a. Bypassing the exponential dependency: Looped transformers efficiently learn in-context by multi-step gradient descent. In *The 28th International Conference on Artificial Intelligence and Statistics*.

Bo Chen, Yingyu Liang, Zhizhou Sha, Zhenmei Shi, and Zhao Song. 2025b. HSR-enhanced sparse attention acceleration. In *The Second Conference on Parsimony and Learning (Proceedings Track)*.

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

Yifang Chen, Xiaoyu Li, Yingyu Liang, Zhenmei Shi, and Zhao Song. 2025c. The computational limits of state-space models and mamba via the lens of circuit complexity. In *Conference on Parsimony and Learning*. PMLR.

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

David Chiang. 2024. Transformers in dlogtime-uniform tc0. *arXiv preprint arXiv:2409.13629*.

Yichuan Deng, Jiangxuan Long, Zhao Song, Zifan Wang, and Han Zhang. 2025. Streaming kernel PCA algorithm with small space. In *The Second Conference on Parsimony and Learning (Proceedings Track)*.

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

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

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

Yiding Hao, Dana Angluin, and Robert Frank. 2022. Formal language recognition by hard attention transformers: Perspectives from circuit complexity. *Transactions of the Association for Computational Linguistics*, 10:800–810.

Adi Haviv, Ori Ram, Ofir Press, Peter Izsak, and Omer Levy. 2022. Transformer language models without positional encodings still learn positional information. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 1382–1390.

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

Ermo Hua, Che Jiang, Xingtai Lv, Kaiyan Zhang, Youbang Sun, Yuchen Fan, Xuekai Zhu, Biqing Qi, Ning Ding, and Bowen Zhou. 2025. Fourier position embedding: Enhancing attention's periodic extension for length generalization. In *Forty-second International Conference on Machine Learning*.

Neil Immerman. 1998. *Descriptive complexity*. Springer Science & Business Media.

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

Amirhossein Kazemnejad, Inkit Padhi, Karthikeyan Natesan Ramamurthy, Payel Das, and Siva Reddy. 2023. The impact of positional encoding on length generalization in transformers. *Advances in Neural Information Processing Systems*, 36:24892–24928.

Guolin Ke, Di He, and Tie-Yan Liu. 2021. Rethinking positional encoding in language pre-training. In *International Conference on Learning Representations*.

Yekun Ke, Xiaoyu Li, Yingyu Liang, Zhenmei Shi, and Zhao Song. 2025. Circuit complexity bounds for visual autoregressive model. *arXiv preprint arXiv:2501.04299*.

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

Tzu-Sheng Kuo, Aaron Lee Halfaker, Zirui Cheng, Jiwoo Kim, Meng-Hsin Wu, Tongshuang Wu, Kenneth Holstein, and Haiyi Zhu. 2024. 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.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.

Xiaoyu Li, Yingyu Liang, Zhenmei Shi, Zhao Song, Wei Wang, and Jiahao Zhang. 2025. On the computational capability of graph neural networks: A circuit complexity bound perspective. *arXiv preprint arXiv:2501.06444*.

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

Yingyu Liang, Jiangxuan Long, Zhenmei Shi, Zhao Song, and Yufa Zhou. 2025. Beyond linear approximations: A novel pruning approach for attention matrix. In *The Thirteenth International Conference on Learning Representations*.

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

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

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

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

William Merrill, Yoav Goldberg, Roy Schwartz, and Noah A Smith. 2021. On the power of saturated transformers: A view from circuit complexity. *arXiv preprint arXiv:2106.16213*.

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

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

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

Meta. 2024. Llama 3.

Swaroop Mishra, Arindam Mitra, Neeraj Varshney, Bhavdeep Sachdeva, Peter Clark, Chitta Baral, and Ashwin Kalyan. 2022. Numglue: A suite of fundamental yet challenging mathematical reasoning tasks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3505–3523.

OpenAI. 2024a. Hello gpt-4o. https://openai.com/index/hello-gpt-4o/. Accessed: May 14.

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

Ofir Press, Noah Smith, and Mike Lewis. 2022. Train short, test long: Attention with linear biases enables input length extrapolation. In *International Conference on Learning Representations*.

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

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

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

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

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

Virginia Vassilevska Williams. 2018. 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.

Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, and 1 others. 2023. The rise and potential of large language model based agents: A survey. *arXiv preprint arXiv:2309.07864*.

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

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

Jiajun Zhu, Peihao Wang, Ruisi Cai, Jason D Lee, Pan Li, and Zhangyang Wang. 2025. Rethinking addressing in language models via contexualized equivariant positional encoding. *arXiv preprint arXiv:2501.00712*.

Appendix

Roadmap. In Section A, we provide some background knowledge about circuit complexity. In Section B, we present some lemmas about floating point number computation. In Section C, we introduce the Arithmetic formula evaluation problem. In Section D, we introduce the Boolean formula value problem. In Section E, we present the missing proofs in Section 4. In Section F, we present additional results for other positional embedding methods.

A Backgrounds about Circuit Complexity

Circuit. In theoretical computer science, a circuit is a collection of interconnected gates that process input data and produce an output (Vollmer, 1999). Each gate in the circuit performs a simple logical operation, such as AND, OR, or NOT. These gates take binary inputs (usually 0 or 1) and produce binary outputs. A circuit can have fanin, which refers to the number of input wires each gate can have, and fanout, which refers to the number of output wires a gate can have. To better understand the construction of the circuit, we introduce uniform and nonuniform. Nonuniform circuit allows for the design of different circuits for inputs of varying sizes. For instance, a circuit built to handle inputs of size n may differ significantly from one designed for inputs of size $n+1$. This approach is flexible, much like how humans only need to provide the existence result of the circuit without worrying about construction algorithms. In contrast, a uniform model requires the construction of these circuits through an algorithm or by solving a different problem within another complexity class, such as DLOGTIME-uniform and L-uniform (as defined in Definition 3.11 and Definition 3.10 of our paper).

Gate. We present details of some common gates here. Constant fan-in Boolean circuit gates represent the most basic building blocks, such as AND, OR, and NOT, strictly limited to a small, fixed number of inputs (typically 2). In contrast, unbounded fan-in Boolean circuit gates relax this restriction, allowing AND and OR gates (though NOT usually remains fan-in 1) to accept an arbitrary number of inputs simultaneously. Distinctly, the MAJORITY gate represents a more powerful computational primitive; it takes multiple inputs and outputs true if and only if a strict majority (more than half) of its inputs are true, thereby performing a threshold computation fundamentally different from simple AND/OR logic and enabling the efficient computation of functions like Parity, which are difficult for standard unbounded fan-in circuits of constant depth.

Circuit Complexity. Circuit is an essential way to model computation because they represent how problems can be solved by performing a series of simple operations, i.e., gates. In the context of circuit complexity, we often care about whether we can solve a problem using a family of circuits. We often use three different things to distinguish different circuit complexity families: the size, the depth, and the types of gate. The size of a circuit refers to the number of gates it contains. The depth of a circuit is the longest path from any input to the output, measured in terms of the number of gates encountered along that path. We will present in detail what is the kind of gate in Different types of gate. If a problem requires circuits with a huge number of gates, depth, or more complex types of gates, we consider it hard. If it can be solved by circuits that are both small in size and shallow in depth, we consider it relatively easy or efficiently parallelize. Previous work (Merrill and Sabharwal, 2023a; Merrill et al., 2022) group problems into different "complexity classes" based on how their circuit size and depth grow as the input size increases.

The Difference between TC^0 and NC^1. Building on circuit concepts like size and depth, we provide the definition of two complexity classes, TC^0 and NC^1, distinguished by their depth limits and gate types (See formal definition in Definition 3.3 and Definition 3.5). NC^1 problems are solvable by polynomial-size circuits with logarithmic depth ($O(\log n)$) using only simple, bounded fan-in Boolean gates (like AND, OR, NOT). In contrast, TC^0 problems use polynomial-size circuits limited to constant depth ($O(1)$) but employ powerful unbounded fan-in threshold gates (like MAJORITY) and potentially unbounded AND/OR gates. Although TC^0 circuits are shallower, their gates are stronger. However, this gate power isn't enough to overcome the depth restriction; TC^0 is contained within NC^1 ($\mathsf{TC}^0 \subseteq \mathsf{NC}^1$, see Fact 3.8). This is because constant-depth threshold gates can be simulated by logarithmic-depth circuits using only bounded fan-in gates, fitting the NC^1 definition. Thus, NC^1 allows greater depth with simple

gates, while TC^0 uses powerful gates restricted to constant depth.

B Floating Point Number Computation

In this section, we first show the definition of some important operations for floating point numbers in Section B.1, and then present the circuit complexity results for floating point operations n Section B.2.

B.1 Basic Operations of Floating Point Numbers

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

Definition B.1 (Rounding, Definition 9 on page 5 of (Chiang, 2024))**.** *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 fundamental concepts in Definition 3.13 and Definition B.1, we can now define the core arithmetic operations needed for Transformer computations:

Definition B.2 (Floating-point number operations, page 5 on (Chiang, 2024))**.** *Let a, b be two integers, we define*

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

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

- *addition:*

$$\langle m_1, e_1\rangle + \langle m_2, e_2\rangle := \begin{cases} r_1 & \text{if } e_1 \geq e_2 \\ r_2 & \text{if } e_1 \leq e_2, \end{cases}$$

  where

$$r_1 := \mathrm{round}_p(\langle m_1 + m_2 /\!\!/ 2^{e_1 - e_2}, e_1\rangle)$$
$$r_2 := \mathrm{round}_p(\langle m_1 /\!\!/ 2^{e_2 - e_1} + m_2, e_2\rangle).$$

- *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:*

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

- *comparison:*

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

- *floor:*

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

B.2 Floating Point Number Operations in TC^0

Lemma B.3 (Standard float point number operations in TC^0, Lemma 10 on page 5 and Lemma 11 on page 6 of (Chiang, 2024))**.** *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 B.2 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 B.4 (Floor operation in TC^0)**.** *Let p be a positive integer. If $p \leq \mathrm{poly}(n)$, then floor operation defined in Definition B.2 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 B.3.*

Proof. This directly follows from the definition of the floor function in Definition B.2. □

Lemma B.5 (Approximating exp in TC^0, Lemma 12 on page 7 of (Chiang, 2024))**.** *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 B.6 (Approximating square root in TC^0, Lemma 12 on page 7 of (Chiang, 2024))**.** *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 $\sqrt{x}$.*

C Arithmetic Formula Evaluation Problem

In this section, we first provide a foundational definition as established in (Buss et al., 1992).

Definition C.1 (Arithmetic formula, Definition on page 13 of (Buss et al., 1992))**.** *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 C.2 (Arithmetic formula evaluation problem, Definition on page 14 of (Buss et al., 1992))**.** *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 $A(c_1, c_2, \cdots, c_n)$?*

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

Lemma C.3 (Theorem 6.1 on page 31 of (Buss et al., 1992))**.** *The arithmetic formula evaluation problem is in* NC^1-complete.

D Boolean Formula Value Problem

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

Definition D.1 (Definition on Page 1 of (Buss, 1987))**.** *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 D.2 (Definition on page 1 of (Buss, 1987))**.** *$|\alpha|$ is the length of α, i.e. the number of symbols in the string α.*

Definition D.3 (Definition on page 1 of (Buss, 1987))**.** *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 D.4 (Page 1 on (Buss, 1987))**.** *The problem of determining the truth value of a Boolean formula is in* NC^1-complete.

E Missing Proofs in Section 4

Here we present some missing proofs in Section 4. We restate Lemma 4.1 below

Lemma E.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 B.3 and Corollary B.4:

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. □

We show the proof of Lemma 4.2 below.

Lemma E.2 (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 B.3, 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 B.3, 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 E.3 (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 3.17 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 3.17.

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 B.5, 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. □

Here we present the proof of Lemma 4.4.

Lemma E.4 (Single RoPE-based attention layer computation in TC^0, formal version of Lemma 4.4)**.** *If $p \leq \mathrm{poly}(n)$, then the attention layer Attn in Definition 3.18 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 B.3. 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 B.3. 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. □

Here we state the proof of Lemma 4.5.

Lemma E.5 (MLP computation in TC^0, formal version of Lemma 4.5)**.** *If $p \leq \mathrm{poly}(n)$, then the MLP layer in Definition 3.20 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 $WX_{i,*}$, and by Part 1 of Lemma B.3, w need a circuit with depth d_{std} and size $\mathrm{poly}(n)$ to compute $WX_{i,*} + b$. 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. □

Here we state the proof of Lemma 4.6.

Lemma E.6 (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 3.21 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 B.3, 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 B.3 and Lemma B.6, 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_{sqrt} and size is $\text{poly}(n)$. Since this procedure can be done in parallel for all $i \in [n]$, the proof is complete. □

Here we state the proof of Lemma 4.7.

Lemma E.7 (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 $\text{poly}(n)$. If $p \leq \text{poly}(n)$, then the RoPE-based Transformer TF in Definition 3.19 can be computable by a uniform threshold circuit with size $\text{poly}(n)$ and depth $(m+1)d_g + 7m(d_{\text{std}} + d_{\oplus}) + m(d_{\triangle} + d_{\exp})$.*

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

For each $i \in [m]$, by Lemma E.4, Attn_i is computable by a uniform threshold circuit with depth $7(d_{\text{std}} + d_{\oplus}) + d_{\triangle} + d_{\exp}$ and size $\text{poly}(n)$.

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 $(m+1)d_g + 7m(d_{\text{std}} + d_{\oplus}) + m(d_{\triangle} + d_{\exp})$ and the size of circuit is $\text{poly}(n)$.

Thus we complete the proof. □

Next, we state the proof for our main results.

Theorem E.8 (Main result, circuit complexity bound of RoPE-based Transformers, formal version of Theorem 4.8)**.** *Suppose that for each $i \in [m]$, g_i in TF is computable by a constant depth d_g uniform threshold circuit with size $\text{poly}(n)$. If $p \leq \text{poly}(n), d \leq O(n), m \leq O(1)$, then the RoPE-based Transformer TF in Definition 3.19 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

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

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

Thus we complete the proof. □

F Results for Additional Positional Embedding Methods

In this section, we present the result that a widely used positional embedding method, Attention with Linear Biases (ALiBi), can be simulated by a uniform TC^0 circuit family. In Section F.1, we present the definition of the ALiBi positional embeddings. In Section F.2, we show the circuit complexity bound of ALiBi Transformers. In Section F.3, we outline the hardness results for ALiBi Transformers.

F.1 The ALiBi Positional Embedding

Following from the positional embedding formulation in (Press et al., 2022), we first define the linear bias matrix, which is the foundation of the ALiBi positional embeddings.

Definition F.1 (Linear bias matrix, implicit in page 5 of (Press et al., 2022))**.** *Let $m \in \mathbb{F}_p$ be a constant. We define the linear bias matrix $B \in \mathbb{F}_p^{n \times n}$ as:*

$$B_{i,j} = -m|i-j|.$$

Differing from previous positional embedding methods (Vaswani et al., 2017; Su et al., 2024), ALiBi *injects the positional embeddings into the attention matrices, instead of the representation of the input tokens. Specifically, its attention matrix can be defined as follows:*

Definition F.2 (ALiBi attention matrix)**.** *Let the linear bias matrix B be defined as Definition F.1. 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* ALiBi *attention matrix $A_{\mathsf{ALiBi}} \in \mathbb{F}_p^{n \times n}$ as*

$$A_{\mathsf{ALiBi}} := \exp(\underbrace{X}_{n \times d} \underbrace{W_Q}_{d \times d} \underbrace{W_K^\top}_{d \times d} \underbrace{X^\top}_{d \times n} + \underbrace{B}_{n \times n}).$$

Remark F.3. *By plugging the* ALiBi *attention matrix (Definition F.2) into Definition 3.18 and Definition 3.19, we obtain the definition for multi-layer* ALiBi *Transformers.*

F.2 Circuit Complexity of ALiBi Transformers

Lemma F.4 (ALiBi-based attention matrix computation in TC^0)**.** *If $p \leq \text{poly}(n)$, then the attention matrix A_{ALiBi} in Definition F.2 can be computable by a uniform threshold circuit with size $\text{poly}(n)$ and depth $4d_{\text{std}} + 3d_{\oplus} + d_{\exp}$.*

Proof. To compute the first term $XW_QW_K^\top X^\top$, we need to perform matrix multiplication three times, where all dimensions are no greater than n. Thus, by applying Lemma 4.2 three times, we

can compute these matrix multiplications in a uniform threshold circuit with depth $3(d_{\text{std}} + d_{\oplus})$ and size $\text{poly}(n)$.

Next, we compute all the element-wise summations between $(XW_Q W_K^\top X^\top)_{i,j}$ and $B_{i,j}$ in parallel for all $i, j \in [n]$. This requires a circuit with depth d_{std} and size $O(n^2) \leq \text{poly}(n)$, following Lemma B.3. Then, we compute the exponential function in depth d_{exp} and size $\text{poly}(n)$, as stated in Lemma B.5.

Combining all the steps above, the total depth of the circuit is:

$$\begin{aligned} d_{\text{total}} &= d_{\text{std}} + 3(d_{\text{std}} + d_{\oplus}) + d_{\text{exp}} \\ &= 4d_{\text{std}} + 3d_{\oplus} + d_{\text{exp}}. \end{aligned}$$

Since each step requires a circuit of size $\text{poly}(n)$, the combined circuit remains within $\text{poly}(n)$ size.

Thus, we complete the proof. □

Theorem F.5 (Circuit complexity bound of ALiBi-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 $\text{poly}(n)$. If $p \leq \text{poly}(n), d \leq O(n), m \leq O(1)$, then the* ALiBi*-based Transformer can be simulated by a uniform TC^0 circuit family.*

Proof. Following Lemma F.4 and similar steps in Lemma E.4, we conclude that the single attention layer $\mathsf{Attn}_i(X)$ in Definition 3.18 with ALiBi positional embedding can be computed using a uniform threshold circuit of $\text{poly}(n)$ size with depth $7d_{\text{std}} + 6d_{\oplus} + d_{\text{exp}}$.

Next, we combine the previous result with similar steps in Lemma E.7 to obtain that the multi-layer ALiBi-based Transformer $\mathsf{TF}(X)$ can be computed using a uniform threshold circuit of $\text{poly}(n)$ size with depth $(m+1)d_g + m(7d_{\text{std}} + 6d_{\oplus} + d_{\text{exp}})$.

Since $m = O(1)$, the circuit depth to compute $\mathsf{TF}(X)$ follows:

$$(m+1)d_g + m(7d_{\text{std}} + 6d_{\oplus} + d_{\text{exp}}) = O(1).$$

Since we also have that the circuit computing $\mathsf{TF}(X)$ has $\text{poly}(n)$ size, we can conclude that the ALiBi-based Transformer can be simulated by a uniform TC^0 circuit family.

The proof can be derived directly from Lemma F.4 and similar steps in Theorem 4.8.

Thus, we complete the proof. □

F.3 Hardness Results of ALiBi Transformers

Since ALiBi-based Transformers has the same circuit complexity bounds as RoPE-based Transformers, the similar hardness results for RoPE-based Transformers also holds for ALiBi-based Transformers. Specifically, we have the following results:

Theorem F.6. *Unless $\mathsf{TC}^0 = \mathsf{NC}^1$, an* ALiBi*-based Transformer with $\text{poly}(n)$-precision, $O(1)$ layers, hidden dimension $d \leq O(n)$ cannot solve the Arithmetic formula evaluation problems.*

Proof. By the hierarchy of circuit families in Fact 3.8, we can obtain that $\mathsf{TC}^0 \subseteq \mathsf{NC}^1$. Since ALiBi-based Transformers are in TC^0 (Theorem F.5) and the This follows from combining Theorem 4.8 (circuit complexity bound of RoPE-base Transformer) and the arithmetic formula evaluation problem is in NC^1 (Lemma C.3), we can conclude that ALiBi-based Transformers cannot solve the arithmetic formula evaluation problem unless $\mathsf{TC}^0 = \mathsf{NC}^1$.

Thus, we complete the proof. □

Theorem F.7. *Unless $\mathsf{TC}^0 = \mathsf{NC}^1$, an* ALiBi*-based Transformer with $\text{poly}(n)$-precision, $O(1)$ layers, hidden dimension $d \leq O(n)$ cannot solve the Boolean formula value problem.*

Proof. By the hierarchy of circuit families in Fact 3.8, we can obtain that $\mathsf{TC}^0 \subseteq \mathsf{NC}^1$. Since ALiBi-based Transformers are in TC^0 (Theorem F.5) and the This follows from combining Theorem 4.8 (circuit complexity bound of RoPE-base Transformer) and the arithmetic formula evaluation problem is in NC^1 (Lemma D.4), we can conclude that ALiBi-based Transformers cannot solve the Boolean formula value problem unless $\mathsf{TC}^0 = \mathsf{NC}^1$.

Thus, we complete the proof. □