Do Efficient Transformers Really Save Computation?

Kai Yang [1] **Jan Ackermann** [2] **Zhenyu He** [3] **Guhao Feng** [1] **Bohang Zhang** [3] **Yunzhen Feng** [4] **Qiwei Ye** [5] **Di He** [3] **Liwei Wang** [3 6]

Abstract

As transformer-based language models are trained on increasingly large datasets and with vast numbers of parameters, finding more efficient alternatives to the standard Transformer has become very valuable. While many efficient Transformers and Transformer alternatives have been proposed, none provide theoretical guarantees that they are a suitable replacement for the standard Transformer. This makes it challenging to identify when to use a specific model and what directions to prioritize for further investigation. In this paper, we aim to understand the capabilities and limitations of efficient Transformers, specifically the Sparse Transformer and the Linear Transformer. We focus on their reasoning capability as exhibited by Chain-of-Thought (CoT) prompts and follow previous works to model them as Dynamic Programming (DP) problems. Our results show that while these models are expressive enough to solve general DP tasks, contrary to expectations, they require a model size that scales with the problem size. Nonetheless, we identify a class of DP problems for which these models can be more efficient than the standard Transformer. We confirm our theoretical results through experiments on representative DP tasks, adding to the understanding of efficient Transformers' practical strengths and weaknesses.

1. Introduction

The Transformer architecture, as introduced in the seminal work of Vaswani et al. (2017), has demonstrated a remarkable performance in numerous applications ranging from natural language processing to computer vision and speech. A significant advancement has recently been made, by scaling up Transformers to build Large Language Models (LLMs) (Brown et al., 2020; OpenAI, 2023; Touvron et al., 2023). These LLMs, exemplified by models like GPT and LLaMa, typically have billions of parameters and are trained on datasets containing trillions of tokens. Given the substantial computational demands, enhancing LLMs' efficiency has become a pivotal research focus in academic and industrial contexts.

The primary computational bottleneck in Transformers arises from the self-attention module, whose complexity scales quadratically with the sequence length. The cost becomes particularly noticeable in tasks that require *long sequence generation*, such as coherent story generation or reasoning with Chain-of-Thought prompts (Wei et al., 2022b; Kojima et al., 2022; Nye et al., 2022; Zhou et al., 2023). Given the practical needs, a large body of work seeks to develop *efficient Transformers* that can reduce the quadratic complexity of self-attention (Tay et al., 2022), typically by imposing sparsity into architectural design (Child et al., 2019; Beltagy et al., 2020; Qiu et al., 2020; Kitaev et al., 2020; Vyas et al., 2020; Roy et al., 2021) or by employing low-rank or kernel-based approximations to accelerate the computation (Katharopoulos et al., 2020; Choromanski et al., 2021; Peng et al., 2021; Wang et al., 2020; Luo et al., 2021). However, there is generally a lack of understanding about the capabilities of efficient Transformer.

In this work, we take a step towards theoretically understanding the capability of efficient Transformers. In particular, we focus on the models' reasoning ability, a fundamental aspect of human intelligence that plays a vital role in problem-solving, decision-making, and planning. Inspired by a recent study in Feng et al. (2023), we model reasoning as a *dynamic programming* (DP) process as it closely resembles the way Chain-of-Thought prompts are executed. The output sequence consists of answers to a series of intermediate steps, each corresponding to solving a subproblem represented by a DP state. Feng et al. (2023) proved that all reasoning problems fitting within this framework can be solved by a standard autoregressive Transformer of a *constant* size (irrelevant to the problem scale), thus achieving a

[1]School of EECS, Peking University [2]ETH Zürich [3]National Key Laboratory of General Artificial Intelligence, School of Intelligence Science and Technology, Peking University [4]New York University [5]Beijing Academy of Artificial Intelligence [6]Center for Machine Learning Research, Peking University. Correspondence to: Di He <dihe@pku.edu.cn>, Liwei Wang <wanglw@pku.edu.cn>, Bohang Zhang <zhangbohang@pku.edu.cn>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

Table 1. Complexity of the Transformer variants on different tasks.

Architecture	General Reasoning	Arithmetic	Reasoning (locality assumption)
Standard Transformer	$\tilde{\Theta}(L^2)$	$\tilde{\Theta}(L^2)$	$\tilde{\Theta}(L^2)$
Sparse Transformer	$\tilde{\Theta}(L^2)$	$\tilde{\Theta}(L\sqrt{L})$	$\tilde{\Theta}(L\sqrt{L})$ if $m = \mathrm{O}(\sqrt{L})$
Linear Transformer	$\tilde{\Theta}(L^2)$	$\tilde{\Omega}(L\sqrt{L})$	$\tilde{\Omega}(mL)$

computational complexity of $\Theta(L^2)$ where L is the length of the output sequence.

In our work, we focus on two representative (Tay et al., 2022) and successful (Tay et al., 2020b; Brown et al., 2020) variants of efficient Transformers: the Sparse Transformer (Child et al., 2019) and the Linear Transformer (Katharopoulos et al., 2020). (In the following we will refer to these two as efficient Transformers.) Our analysis shows that both architectures possess the necessary expressiveness for all problems within this DP framework despite only scaling with $\Theta(L\sqrt{L})$ and $\Theta(L)$. Although this positive result might lead one to believe that we can supplant the standard Transformer with others of lower complexity, the situation is more complicated: our main result highlights that both Sparse Transformer and Linear Transformer require a *growing* model size with respect to the problem scale L, in contrast to the *constant* size of standard Transformers. Specifically, under mild assumptions, we prove that neither architecture can generate the DP solution unless the hidden dimension of the network layers scales as $\tilde{\Omega}(\sqrt{L})$. This scaling results in a total computational complexity of $\tilde{\Omega}(L^2)$, matching the vanilla Transformer's complexity. But this result introduces a paradox: when tackling general DP problems, the touted efficiency of these "efficient" Transformers appears to dissolve, rendering them comparably efficient to standard Transformers.

The above findings about general DP problems raise the question: For which problems are efficient Transformers efficient? To answer this question, we start by studying a fundamental task for reasoning: evaluating arithmetic expressions (Feng et al., 2023). Notably, we find that the complexity lower bound can be improved to $\tilde{\Omega}(L\sqrt{L})$ for both architectures, and the lower bound can be attained for the Sparse Transformer with a constant hidden dimension. Motivated by this finding, we then identify a general condition that unlocks the efficiency of efficient Transformers, called the *locality* assumption. Intuitively, this assumption states that each step in the reasoning process only depends on the outcome of *recent* m reasoning steps where m is far smaller than L, i.e. $m = o(L)$. Under this assumption, we show that the complexity lower bound can be improved for sparse- and Linear Transformer. We summarize our theoretical results in Table 1.

We complement our theoretical findings with an extensive set of experiments. Following Feng et al. (2023), we focus on the Arithmetic task and two additional DP problems: the Longest Increasing Subsequence (LIS) and the Edit Distance (ED). Notably, the ED task satisfies the locality assumption, whereas the LIS task does not. For each task, we systematically investigate how variations in the problem size (i.e., the sequence length L) and the hidden dimension of the Transformer models impact the models' performance. Empirical evidence confirms that, for both efficient Transformers, the required hidden dimension increases as the problem size grows in most scenarios, while this is not the case for the standard Transformer. Moreover, the dependency between hidden dimension and problem scale is more pronounced in LIS than in ED. These results validate our theory and offer practical insights into the strengths and weaknesses of efficient Transformers.

Notations. We adopt the big-O notation throughout this paper. Specifically, given two functions $f, g : \mathcal{X} \to [0, \infty)$ where $\mathcal{X}$ can be any set, we write $f = \mathrm{O}(g)$ if there exists a constant $c > 0$ such that $f(x) \leq cg(x)$ for all $x \in \mathcal{X}$. We also write $f = \Omega(g)$ if $g = \mathrm{O}(f)$, and write $f = \Theta(g)$ if both $f = \mathrm{O}(g)$ and $f = \Omega(g)$ hold. Moreover, given two functions $f, g : \mathbb{N}_+^d \to [0, \infty)$, we write $f = \tilde{\mathrm{O}}(g)$ if there exist constants $c, k > 0$ such that $f(x) \leq cg(x) \log^k(x_1 \cdots x_d)$ for all $x \in \mathbb{N}_+^d$. The notations $\tilde{\Omega}(\cdot)$ and $\tilde{\Theta}(\cdot)$ can be similarly defined.

2. Related Work

Transformers and Large Language Models have received significant attention due to their unprecedented success across various domains. A considerable body of literature has emerged to establish a deeper theoretical understanding of their strengths and constraints.

Universal Approximation. Initially, the theoretical focus was on the capacity of Transformers to approximate diverse functions. Yun et al. (2019) postulated that adequately sized Transformers can universally approximate any continuous sequence-to-sequence functions within certain bounds. A parallel line of work first showed that Transformers with infinite precision are turing-complete (Pérez et al., 2019; 2021) and later Wei et al. (2022a) established that Transformers with finite precision are approximately turing-complete. Recently, Alberti et al. (2023) proved that Linear Transformers are also universal approximators. Whereas these results approach expressiveness by proving computational capacity, we complement our expressiveness results with complexity lower bounds for practical settings.

Formal Language Learning. Additionally, the Transformer's expressivity has been studied in the context of

formal language learning. Bhattamishra et al. (2020) constructed a Transformer that detects counter languages, and Yao et al. (2021) show how to detect Dyck languages. Liu et al. (2022) show that shallow Transformers can learn finite state automata and simulate them for a number of steps that scale with the model size. Conversely, Hahn (2020) shows that transformers can not learn distributions over languages. Other works use classical techniques from circuit complexity (Furst et al., 1984) to prove that Transformers can simulate classes of circuits (Hao et al., 2022; Merrill et al., 2022; Merrill & Sabharwal, 2023).

Measuring Complexity. Weiss et al. (2021) introduce a programming language that maps to learnable Transformer encoders and facilitates the analysis of the complexity of problems with respect to layers and attention heads. Sanford et al. (2023) introduce a sparse averaging task that requires recurrent and feed-forward networks to be of linear complexity, whereas the Transformer only needs to scale logarithmically. These works are similar to ours in that we establish concrete relationships between model complexity and solvability of the posed problems. But our work deals with autoregressive efficient Transformers equipped with Chain-of-Thought.

In-context learning. A recent approach shows its in-context learning ability (Garg et al., 2022; Brown et al., 2020). Following this, there are also theoretical results that (Dai et al., 2023; Von Oswald et al., 2023; Akyürek et al., 2022) prove it can perform gradient descent. Another line of work shows in-context-learning via induction heads (Elhage et al., 2021; Olsson et al., 2022). Similarly, Feng et al. (2023) show that auto regressive transformers can learn to perform dynamic programming when equipped with Chain-of-Thought. While in the same setting as Feng et al., we investigate efficient Transformers and present a problem class that encourages efficiency.

Efficient Transformer. Due to the high complexity of the attention layer, many more efficient methods have been proposed. A first series of ideas exploit fixed attention patterns (Child et al., 2019; Beltagy et al., 2020; Qiu et al., 2020). Another line of work approximates the attention as a low rank matrix or with kernels (Katharopoulos et al., 2020; Wang et al., 2020; Choromanski et al., 2021) and further works deal with learned patterns (Kitaev et al., 2020; Tay et al., 2020a; Roy et al., 2021). A last set of works even completely move away from transformers (Sun et al.; Gu & Dao, 2023). Two recent works study when standard attention can be efficient (Alman & Song, 2023) and how to approximate standard attention in linear time (Keles et al., 2023). In contrast to their work, we give theoretical analyses for existing and popular efficient Transformers.

3. Efficient Transformers

The autoregressive Transformer, also called the decoder-only Transformer (Radford et al., 2019; Dai et al., 2019), is a sequence-to-sequence neural network defined as follows. Given an input sequence $\boldsymbol{s}$ of length n, it first transforms each input token s_i ($i \in [n]$) into a D-dimensional vector $\boldsymbol{x}^{(0)} = \mathrm{Embed}(s_i) + \boldsymbol{p}_i \in \mathbb{R}^D$, where $\mathrm{Embed}(\cdot)$ is the token embedding layer and $\boldsymbol{p}_i$ a learnable positional embedding. Then, M Transformer blocks follow, the l-th of which has the following form:

$$\boldsymbol{h}_i^{(l)} = \boldsymbol{x}_i^{(l-1)} + \mathrm{Attn}^{(l)}(\boldsymbol{x}_i^{(l-1)}; \{\boldsymbol{x}_j^{(l-1)} : j \in [i]\}), \quad (1)$$

$$\boldsymbol{x}_i^{(l)} = \boldsymbol{h}_i^{(l)} + \mathrm{FFN}^{(l)}(\boldsymbol{h}_i^{(l)}) \quad (2)$$

Here, $\mathrm{Attn}^{(l)}$ and $\mathrm{FFN}^{(l)}$ denote the multi-head self-attention layer and the feed-forward network of the l-th Transformer block, respectively:

$$\mathrm{Attn}^{(l)}(\boldsymbol{x}, \mathcal{S}) = \sum_{h=1}^{H} \left(\boldsymbol{W}_{\mathrm{O}}^{(l,h)}\right)^{\top} \cdot \mathrm{H}^{(l,h)}(\boldsymbol{x}, \mathcal{S}), \quad (3)$$

$$\mathrm{H}^{(l,h)}(\boldsymbol{x}, \mathcal{S}) = \frac{\sum_{\boldsymbol{z} \in \mathcal{S}} \exp\left((\boldsymbol{W}_{\mathrm{K}}^{(l,h)}\boldsymbol{z})^{\top}(\boldsymbol{W}_{\mathrm{Q}}^{(l,h)}\boldsymbol{x})\right)\boldsymbol{W}_{\mathrm{V}}^{(l,h)}\boldsymbol{z}}{\sum_{\boldsymbol{z} \in \mathcal{S}} \exp\left((\boldsymbol{W}_{\mathrm{K}}^{(l,h)}\boldsymbol{z})^{\top}(\boldsymbol{W}_{\mathrm{Q}}^{(l,h)}\boldsymbol{x})\right)}, \quad (4)$$

$$\mathrm{FFN}^{(l)}(\boldsymbol{x}) = \boldsymbol{W}_2^{(l)}\sigma(\boldsymbol{W}_1^{(l)}\boldsymbol{x}), \quad (5)$$

where $\boldsymbol{W}_{\mathrm{Q}}^{(l,h)}, \boldsymbol{W}_{\mathrm{K}}^{(l,h)}, \boldsymbol{W}_{\mathrm{V}}^{(l,h)}, \boldsymbol{W}_{\mathrm{O}}^{(l,h)} \in \mathbb{R}^{\lceil \frac{D}{H} \rceil \times D}$ are the query, key, value, output matrices of the h-th head in the l-th layer, respectively, and $\boldsymbol{W}_1^{(l)}, \boldsymbol{W}_2^{(l)} \in \mathbb{R}^{D \times D}$ are weight matrices in the FFN. The activation σ is chosen as GeLU (Hendrycks & Gimpel, 2016), following (Radford et al., 2019; Devlin et al., 2019). The computed embedding $\boldsymbol{x}_n^{(M)}$ will be used to predict the next token s_{n+1}, which is then concatenated to the input to continue the sequence generation process. The process stops when an `End-of-Sentence` token is generated.

Based on Equations (1) to (3) and (5), it is easy to see that the computational complexity of an autoregressive Transformer is $\Theta(M(L^2D + LD^2))$, where L is the sequence length. This quadratic dependency on L limits the application of Transformers to long text, in particular for complex reasoning tasks. To battle this, researchers have proposed various efficient Transformers to reduce the complexity. In our work, we investigate the Sparse Transformer and the Linear Transformer. Below, we describe the two architectures which are studied in this paper.

Sparse Transformer. Unlike the standard Transformer where each token $\boldsymbol{x}^{(l)}$ can attend to all previous positions $\{\boldsymbol{x}_j^{(l)} : j \in [i]\}$ (see Equation (1)), in a Sparse Transformer it only attends to a *subset* of previous tokens $\{\boldsymbol{x}_j^{(l)} : j \in \mathcal{I}_i\}$.

In this paper, we study a standard design paradigm proposed in Child et al. (2019), which employs a block-wise pattern as shown in the following:

$$\mathcal{I}_i = \{j : i - kB < j \le i\} \cup \{j : j - 1 \bmod B \ge B - c\} \tag{6}$$

where B is called the block size and k, c are constant integers. When $B = \Theta(\sqrt{L})$, the Sparse Transformer achieves a minimal complexity of $\Theta(M(L\sqrt{L}D + LD^2))$. We note that GPT-3 adopted the above design paradigm (Brown et al., 2020).

Linear Transformer. Another line of work proposed to accelerate the attention computation (Equation (3)) using kernel-based approximations. A representative approach is the Linear Transformer (Katharopoulos et al., 2020), which approximates $\mathrm{Attn}^{(l)}$ with the following formula:

$$\mathrm{Attn}^{(l)}_{\text{linear}}(\boldsymbol{x}, \mathcal{S}) = \sum_{h=1}^{H} \left(\boldsymbol{W}_{\mathrm{O}}^{(l,h)}\right)^{\top} \cdot H^{(l,h)}_{\text{linear}}(\boldsymbol{x}, \mathcal{S}), \tag{7}$$

$$H^{(l,h)}_{\text{linear}}(\boldsymbol{x}, \mathcal{S}) = \frac{\sum_{\boldsymbol{z}\in\mathcal{S}} \phi(\boldsymbol{W}_{\mathrm{K}}^{(l,h)}\boldsymbol{z})^{\top}\phi(\boldsymbol{W}_{\mathrm{Q}}^{(l,h)}\boldsymbol{x})(\boldsymbol{W}_{\mathrm{V}}^{(l,h)}\boldsymbol{z})}{\sum_{\boldsymbol{z}\in\mathcal{S}} \phi(\boldsymbol{W}_{\mathrm{K}}^{(l,h)}\boldsymbol{z})^{\top}\phi(\boldsymbol{W}_{\mathrm{Q}}^{(l,h)}\boldsymbol{x})} \tag{8}$$

where they choose $\phi(\boldsymbol{x}) = \mathrm{elu}(\boldsymbol{x}) + \mathbf{1}$. The above computation can be accelerated by rearranging the order of computation so that the intermediate results $\sum_{\boldsymbol{z}\in\mathcal{S}}(\boldsymbol{W}_{\mathrm{V}}^{(l,h)}\boldsymbol{z})\phi(\boldsymbol{W}_{\mathrm{K}}^{(l,h)}\boldsymbol{z})^{\top}$ and $\sum_{\boldsymbol{z}\in\mathcal{S}} \phi(\boldsymbol{W}_{\mathrm{K}}^{(l,h)}\boldsymbol{z})^{\top}$ associated with different $\mathcal{S}$ can be jointly computed using prefix sum, finally yielding a complexity of $\Theta(MLD^2)$ which is linear in L.

4. Expressiveness of Efficient Transformers in Reasoning Tasks

Reasoning constitutes a fundamental aspect of human intelligence and plays a vital role in problem-solving, decision-making, and planning. Recently, Transformer-based LLMs have demonstrated remarkable reasoning abilities (OpenAI, 2023; Touvron et al., 2023). This has sparked a series of studies aimed at theoretically understanding how powerful these models are. In particular, Feng et al. (2023) recently revealed that autoregressive Transformers are capable of solving a general class of reasoning problems formalized as Dynamic Programming (DP). In this section, we extend this finding by investigating how things change when moving to various types of efficient Transformers.

4.1. Problem formulation

Dynamic programming decomposes a complex reasoning problem into a sequence of reasoning steps, each of which corresponds to a subproblem and is called a DP state. Different subproblems depend on each other because they can be efficiently solved based on the answers of previously solved subproblems. Formally, denoting by $\mathsf{dp}(i)$ the answer of subproblem i, then the relation between subproblems can be characterized using a transition function:

$$\mathsf{dp}(i) = f\left(i, \mathsf{dp}(h_1(i)), \cdots, \mathsf{dp}(h_K(i)), s_{g_1(i)}, \cdots, s_{g_J(i)}\right) \tag{9}$$

where $\boldsymbol{s}$ is the input sequence, and f, $g_1, \cdots, g_J$, $h_1, \cdots, h_K$ are functions that depends on the problem. In other words, the answer of each subproblem is fully determined by the answers of a finite number of previous subproblems plus a finite number of input tokens. Based on Equation (9), we can sequentially solve all subproblems one by one. After solving all subproblems, the final answer can be computed by $u(\mathsf{dp}(i_N))$, where i_N is the last DP state and u is a problem-dependent function. By defining our problem so generally, we also cover CoT problems. We assume that the f, $\boldsymbol{g}$, $\boldsymbol{h}$ and u above can be approximated by an MLP with GeLU activation of constant size. We also assume that during the CoT generation process, the next state can be obtained by an MLP where the input is the current state. One can refer to Appendix B for a formal description. We argue that these assumptions are mild and that they have been used in previous work (Feng et al., 2023).

In our subsequent analysis, without loss of generality, we assume that each input element s_j is an integer, and each state i, DP value $\mathsf{dp}(i)$, and the final answer can all be represented by vectors of integer elements. The domain of these integers can grow polynomially with respect to the length L.

Output format. Following Feng et al. (2023), given a DP task and an input seuqence $\boldsymbol{s}$, an autoregressive Transformer generates the answer with all intermediate steps in the following form:

$$\begin{aligned}&(s_1, \mathbf{0}, \mathbf{0}, \mathbf{0}) \quad \dots \quad (s_n, \mathbf{0}, \mathbf{0}, \mathbf{0}) \quad |\\ &(0, i_1, \mathsf{dp}(i_1), \mathbf{0}) \quad \dots \quad (0, i_N, \mathsf{dp}(i_N), \mathbf{0})\\ &(0, \mathbf{0}, \mathbf{0}, u(\mathsf{dp}(i_N)))\end{aligned} \tag{10}$$

Here, the subsequence ending at the special token "|" is the input to the Transformer, and the remainder will be autoregressively generated. The output at each position is split into four parts that store the input, state, DP value, and final answer, respectively. We denote by $i_1, \cdots, i_N$ the sequence of DP states representing all subproblems in order. We consider the regression setting where the output at each position is simply obtained from the embedding of the last Transformer layer by projecting each dimension to the nearest integer. Similarly, each generated output directly serves as the input of the next position (without using a token embedding layer).

Log-precision Transformers. We adopt a realistic and widely-used setting where all internal neurons in the Transformer can only store floating-point numbers within a finite

$O(\log L)$ bit precision (Merrill & Sabharwal, 2023; Liu et al., 2023; Feng et al., 2023), and all basic floating-point computations are truncated, as implemented on a computer. Log-precision implies that each neuron has a limited capacity for computation and information storage. Nevertheless, they remain powerful as they can represent a large range of values (i.e., polynomial in the sequence length L), recovering important quantities like positional embedding.

Under the above assumptions, Feng et al. (2023) proved the following main result for the standard transformer:

Theorem 4.1 (informal)**.** *Consider any DP problem defined above that satisfies the assumptions from B. For any integer $n > 0$, there exists a log-precision autoregressive Transformer with a constant depth M, a constant hidden dimension D, and a constant number of attention heads H (independent of n) that can generate the correct output for all inputs $\boldsymbol{s}$ of length n.*

4.2. Main results

Now, we investigate whether the efficient Transformers defined in Section 3 are as powerful as the standard Transformer in solving DP problems. In particular, we establish a similar result to Theorem 4.1, which we present in the theorem below:

Theorem 4.2. *Consider any DP problem satisfying the same condition as in Theorem 4.1. Given any integer $n > 0$, let L be the length of the output sequence when the input sequence length is n. Then, for both (log-precision) Sparse Transformer with block size $B = \Theta(\sqrt{L})$ and Linear Transformer, there is a model with a constant depth M, a constant number of attention heads H, and a hidden dimension $D = \mathrm{O}(\sqrt{L})$ that can generate the correct output for all inputs $\boldsymbol{s}$ of length n.*

The proof of Theorem 4.2 is non-trivial and is deferred to Appendix B.1. In the proof, we give explicit constructions of parameters for sparse/linear attention and FFN layers, showing that these layers can implement a set of basic operations presented in Appendix A. We then use these operations as building blocks to form a complete model that solves the DP task.

Theorem 4.2 suggests that replacing the standard self-attention with these efficient variants does not restrict the model's expressiveness in reasoning. However, when we compare the total complexity $\mathrm{O}(L^2)$ of the derived models, we can see that it is the same as for the standard attention, which only needed $d = \mathrm{O}(1)$.

Is the increase in model size necessary? Theorem 4.2 only gives a complexity *upper bound* for Sparse/Linear Transformers. It remains to show whether the bound is *tight* and what the *lower bound* of the required hidden dimension is. To answer this question, we will focus on a restricted class of DP problems which we call *regular* DP problems:

Definition 4.3. A DP problem is called *regular* if for any two different input sequences $\boldsymbol{s}^{(1)}$ and $\boldsymbol{s}^{(2)}$ (of the same length) and a fixed but arbitrary model that solves the DP problem, there is a state i such that $\mathsf{dp}(i)$ is different between input $\boldsymbol{s}^{(1)}$ and $\boldsymbol{s}^{(2)}$.

We remark that regularity is a weak assumption, which only states that the reasoning process (not the final answer) should not be exactly the same when the input changes. For example, it excludes the case where the whole DP process does not depend on a specific input element s_j. Equipped with the regularity assumption, we present a central impossibility result:

Theorem 4.4. *Consider any regular DP problem satisfying the same condition as in Theorem 4.1. Assume that the output sequence length L is proportional to the input sequence length n, i.e., $L = \Theta(n)$. Then, given a sufficiently large n, for both (log-precision) Sparse Transformer with block size $B = \Theta(\sqrt{L})$ and Linear Transformer, a model with a constant depth M and a constant number of attention heads H can generate the correct output for all inputs $\boldsymbol{s}$ of length n only if the hidden dimension $D = \tilde{\Omega}(\sqrt{L})$.*

As presented in Appendix B.2, the proof of Theorem 4.4 is based on the following finding: there are inherent information *bottlenecks* in both types of efficient Transformers. Here, the bottleneck is a set of neurons whose values completely determine all ensuing outputs from a specific position. Due to the log-precision assumption, these neurons only store a limited amount of information. Hence it is only possible to recover all subsequent outputs when the hidden dimension is $\tilde{\Omega}(\sqrt{L})$ — otherwise, the Pigeonhole principle will imply that there are two different input sequences that share the same set of neuron values, yielding the contradiction by Definition 4.3.

5. When Can Efficient Transformers Really Save Computation?

In the previous section, we showed the surprising result that these efficient Transformers may not lead to reduced complexity compared to the standard Transformers in general reasoning tasks. However, one should not hastily jump to the conclusion that these efficient Transformers are always inefficient. In this section, we will discuss in which situations efficient Transformers are efficient.

5.1. A motivating example: evaluating arithmetic expressions

We begin by investigating a less complex task proposed by Feng et al. (2023), called the arithmetic evaluation. The task is to evaluate an arithmetic expression like "$2 \times (1 + 5) \div 4$"

and the complete output sequence looks like "$2 \times (1+5) \div 4 = 2 \times 6 \div 4 = 12 \div 4 = 3$". Feng et al. (2023) proved that a standard Transformer of a constant size can solve this task. Surprisingly, we find that a similar result also holds for a constant-size Sparse Transformer, as shown in the proposition below:

Proposition 5.1 (informal)**.** *For any integer n, there exists a log-precision Sparse Transformer with block size $B = \Theta(\sqrt{L})$, 5 layers, 5 attention heads per layer, a constant hidden dimension that can generate the correct output for the arithmetic evaluation task for all expressions of length no more than n.*

Owing to the constant dimension, the complexity of the arithmetic evaluation task can be reduced to $\mathrm{O}(L\sqrt{L})$ by using Sparse Transformers. We next turn to Linear Transformers. While we do not give explicit constructions of model parameters that can solve the arithmetic task, one can still derive a lower bound by using a similar analysis as in Theorem 4.4:

Proposition 5.2 (informal)**.** *For any integer n, a log-precision Linear Transformer with a constant depth M and a constant number of heads H can generate the correct output for the arithmetic evaluation task for all expressions of length no more than n only if the hidden dimension $D = \tilde{\Omega}(\sqrt[4]{L})$.*

Based on this result, the complexity lower bound of Linear Transformers scales like $\tilde{\Omega}(L\sqrt{L})$, which, interestingly, matches that of Sparse Transformers and is also strictly less than $\tilde{\Omega}(L^2)$. This finding naturally raises the following question: Why do these efficient Transformers no longer require a large hidden dimension on the arithmetic task? We will answer this question in the next subsection.

5.2. Locality encourages efficiency

A key difference of the arithmetic task compared to general DP is that its reasoning process exhibits inherent *structures*. To be specific, the output sequence of arithmetic computation can be partitioned into blocks (separated by the symbol "="), where the content of each block depends solely on the preceding one and is irrelevant to other historical blocks. This paradigm is often named as data locality in computer science literature and is also common in general reasoning processes that follow the so-called Chain of Thought format (Wei et al., 2022b). In light of this, we consider a special class of DP problems dubbed the m-locality DP, which is formally defined below:

Definition 5.3 (m-locality DP)**.** Consider a DP problem with output sequence $\boldsymbol{o}_1, \cdots, \boldsymbol{o}_L$ of the form (10) where $\boldsymbol{o}_1, \cdots, \boldsymbol{o}_n$ is the input sequence. The DP problem is said to satisfy the m-locality condition for some $m \geq n$, if there exist functions $f, h_1, \cdots, h_K$ such that for all $i \in [L]$, $\boldsymbol{o}_i = f(\boldsymbol{o}_{h_1(i)}, \cdots, \boldsymbol{o}_{h_K(i)})$, where $i - m \leq h_k(i) < i$ for $k \in [K]$.

In other words, the m-locality condition simply says that each DP state only depends on recent m DP states. Note that the assumption $m \geq n$ is necessary to ensure that all inputs contribute to the answer of the DP problem. Below, we will discuss how the required hidden dimension of efficient Transformers can be reduced when m is far smaller than L.

We first consider the Sparse Transformer, where we have the following result:

Proposition 5.4. *Consider any m-locality DP problem satisfying the same condition as in Theorem 4.1. Given any integer $n > 0$, let L be the length of the output sequence when the input sequence length is n. Then, there exists a (log-precision) Sparse Transformer with block size $B = \Theta(m)$, a constant depth M, a constant number of attention heads H, and a constant hidden dimension D that can generate the correct output for all inputs $\boldsymbol{s}$ of length n.*

As a result, the complexity of Sparse Transformer scales like $\tilde{\mathrm{O}}(mL)$, which is strictly less than $\tilde{\Theta}(L^2)$ when m is far smaller than L. We next turn to the Linear Transformer, where we have the following lower bound:

Proposition 5.5. *Consider any m-locality regular DP problem satisfying the same condition as in Theorem 4.1 and assume that $m = \Theta(n)$ where n is the input sequence length. Then, a log-precision Linear Transformer with a constant depth M and a constant number of heads H can generate the correct output for all inputs $\boldsymbol{s}$ of length n only if the hidden dimension $D = \tilde{\Omega}(\sqrt{m})$.*

The above result implies that the complexity lower bound of Linear Transformer, which is imposed by the bottleneck, scales like $\tilde{\Omega}(mL)$, which is strictly less than $\tilde{\Theta}(L^2)$ when m is far smaller than L. However, we remark that it remains a challenging open question of whether such a complexity lower bound can be matched.

6. Experiments

In the preceding sections, we conducted a theoretical analysis to assess the capabilities of efficient Transformers for general DP problems and problems with locality. This section serves to validate those findings through comprehensive empirical experimentation. Inspired by the reasoning evaluation in (Feng et al., 2023), we adopt a similar experimental design using common DP problems with Chain-of-Thought demonstrations. We focus on understanding how two key factors, problem size and the embedding dimension of the Transformer model, affect performance across tasks.

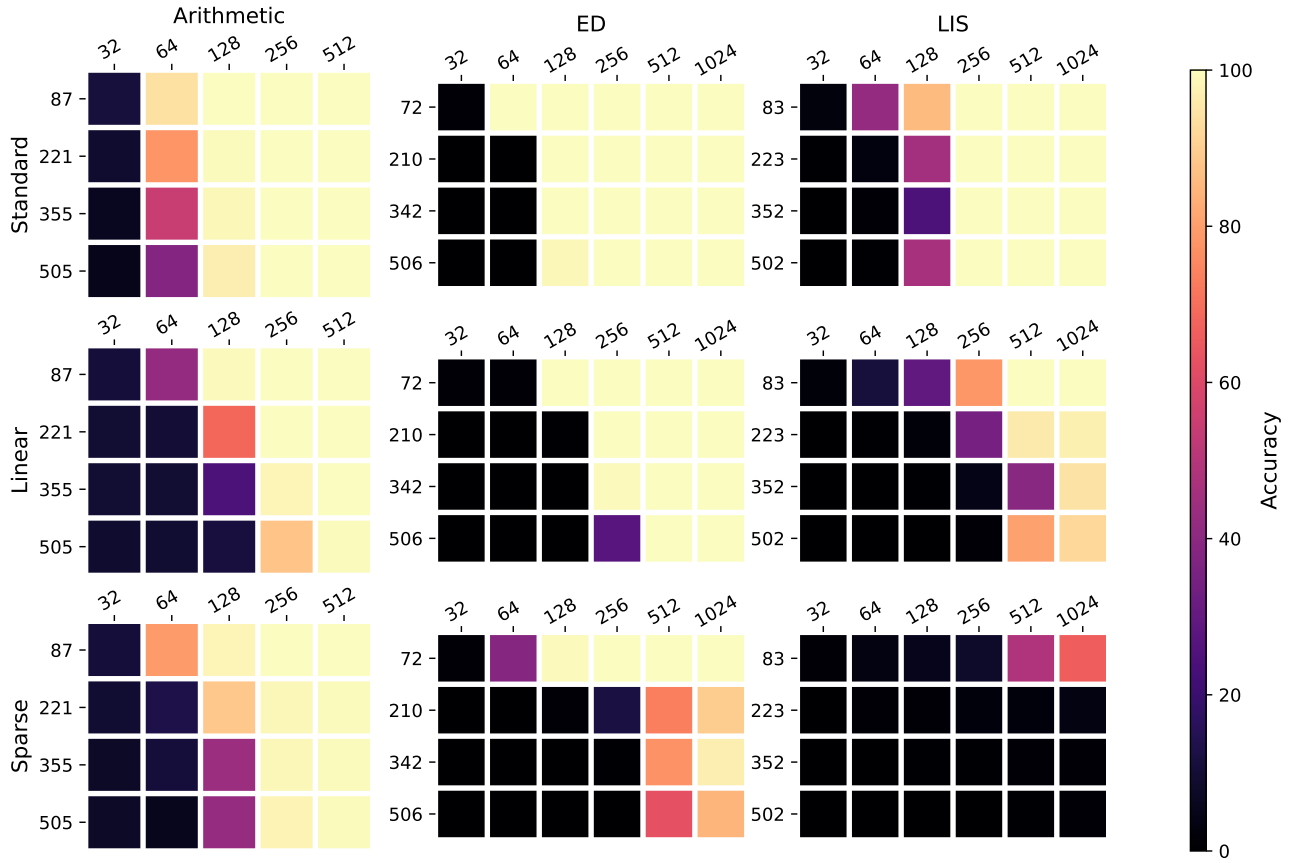

Figure 1. A comparison of accuracies across different tasks and model types. Each column corresponds to a task (Arithmetic, ED, LIS), and each row to a model (Standard Transformer, Linear Transformer, Sparse Transformer). Within each subplot, the x-axis represents the embedding dimension, and the y-axis denotes the problem size. The color intensities indicate the accuracy level achieved by the respective models. The figure demonstrates that efficient Transformers need larger hidden dimensions and that this requirement increases with problem size. It also highlights how standard Transformers can handle tasks across all difficulty levels with fixed embedding dimensions.

6.1. Experimental Design

Tasks and datasets. We chose three well-known tasks – LIS, ED, and Arithmetic – to represent a variety of problems in the DP domain. For the LIS task, the goal is to find the length of the longest increasing subsequence of a given integer sequence. The ED task's goal is to calculate the minimum cost required to convert one sequence to another using three basic edit operations: insert, delete, and replace. For the Arithmetic task, the goal is to calculate the correct result of an arithmetic expression consisting of numbers, addition, subtraction, multiplication, division, and brackets. The LIS task is the most general DP problem without locality property, while ED and Arithmetic exhibit higher locality.

Following previous work (Feng et al., 2023), we curate five datasets for each task, with different problem sizes and increasing difficulty. For the LIS task, the datasets encompass sequences of lengths $\{40, 110, 175, 250\}$, equating to CoT lengths of $\{83, 223, 353, 503\}$. For the ED task, the datasets span varying sequences of lengths – specifically, of averaged lengths 6, 12, 16, and 20, equating to maximum CoT lengths of $\{72, 210, 342, 506\}$. For the Arithmetic task, the dataset consists of sequences with operator numbers in $\{6, 10, 13, 16\}$, equating to maximum CoT lengths of $\{87, 221, 355, 505\}$. Each training dataset has 1M samples, and each corresponding testing dataset has 0.1M.

Model configurations. For the standard Transformer model, we use the same configurations as used by (Feng et al., 2023) with 3 layers and 4 attention heads, albeit with varying embedding dimensions. We employ sinusoidal positional embeddings and apply Xavier initialization to all parameters. For the activation function, we chose the standard GeLU, and the embedding dimensions for the ED and LIS tasks span the range $32, 64, 128, 256, 512, 1024$ uniformly. As for the Arithmetic task, we exclude the 1024 embedding dimensions as all the models have already performed well in 512. The FFN layer's hidden dimension is four times the embedding dimension. We use the same configurations for both the Linear and Sparse Transformers. Within the Sparse Transformer, the block size B is $2^{\lfloor \log_2(\sqrt{L}) \rfloor}$, with L representing the upper limit of CoT length. Every experi-

Table 2. Minimum GFLOPs of different model types to achieve an accuracy above 90% on Arithmetic Task.

Method	Length 87	Length 221	Length 355	Length 505
Standard	0.027	0.266	0.426	0.605
Linear	0.422	4.220	6.756	9.584
Sparse	0.106	1.055	1.690	2.399

ment has the global token count c fixed at 1 for every block. Furthermore, we conduct the experiments on Transformer models with 5 layers while keeping other parameters the same. The additional results are shown in Appendix D.

Model training and inference. In all experiments, we employ the AdamW optimizer (Loshchilov & Hutter, 2017) with the following hyperparameters: $\beta_1 = 0.9, \beta_2 = 0.999, lr = 10^{-4}$, and weight decay $= 0.01$. To enhance model generalization, we maintain a consistent dropout rate of 0.1. Our optimization minimizes the negative log-likelihood loss for all tokens in both the CoT steps and the answers. Each model does 100 training epochs with a batch size of 512. During the inference stage, the model generates the entire CoT process token by token, using greedy search until reaching the `End-of-Sentence` token. We evaluate the models' performance using the accuracy of the final answer, which is the last output in the sequence. We run all experiments on four V100 GPUs.

6.2. Experimental Results

Figure 1 shows our main results. Each column of the figure corresponds to a particular task, and each row to a different model. Within each subplot, the x-axis shows the embedding dimension, and the y-axis indicates the problem size. The color intensities indicate the corresponding accuracies.

For almost all tasks and varying problem sizes, models with sufficiently large embedding dimensions can achieve nearly 100% accuracy, except for the LIS task with the Sparse Transformer. Nevertheless, for this task, the accuracy still increases as the embedding dimension increases. This finding shows that efficient Transformers can handle these DP tasks with adequate expressiveness.

When we compare the subplots column-wise, it is evident that efficient Transformers generally need larger hidden dimensions than standard Transformers. Moreover, within each subplot of efficient Transformers, the required embedding dimension increases as the problem grows. In contrast, standard Transformers, with fixed embedding dimensions of 128 or 256, can handle tasks across all difficulty levels. These observations confirm our theoretical findings, suggesting that efficient Transformers are less efficient than previously perceived. In Table 2, we further compare the minimal number of FLOPs required to achieve 90% accuracy on Arithmetic for all models. The standard Transformer requires the lowest number of flops across all lengths.

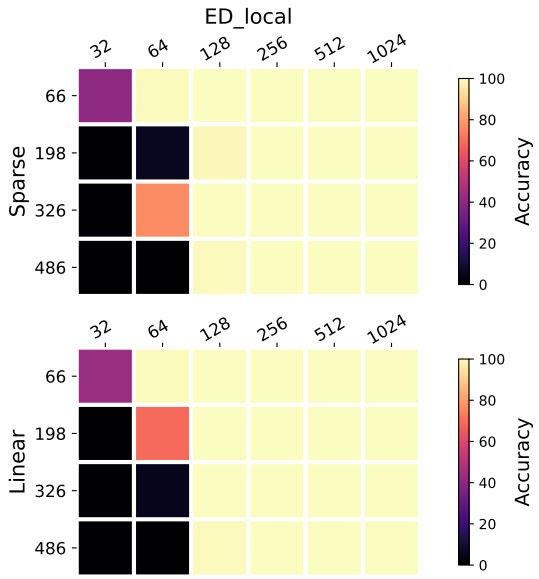

Figure 2. Accuracies of the Sparse Transformer and Linear Transformer on the ED_Local task with varying problem size on the y- and embedding dimension on the x-axis. We can observe that both models benefit from locality.

Comparing the subplots row-wise, the growth in required embedding dimension with the model size becomes more pronounced as the locality decreases. This suggests that efficient Transformers are more efficient for DP tasks with strong locality, which aligns with our previous results.

Locality Study. Although the previous results already indicate that problems with higher locality require higher embedding dimensions, we conducted another more explicit experiment in which we modified the ED problem to have a higher locality. We call this problem ED_Local and show the results in Figure 2. We can clearly see that the increased locality led to a reduction in necessary embedding size, which backs up our theoretical findings.

7. Limitations & Conclusion

Limitations. Although we show our results for representative efficient Transformers, it does not mean that our findings directly transfer to all models with similar designs. Further, despite our experiments indicating that Linear Transformers also benefit from locality, it remains to prove whether the bound for the Linear Transformer can be tightened.

Conclusion. While the Sparse Transformer and Linear Transformer are expressive enough to solve general DP tasks, our findings indicate that they can not sustain their efficiency in the general case. This contradicts the anticipated efficiency gains, bringing their performance closer to that of standard Transformers, which maintain a constant model size. The paradoxical nature of these efficient Transformers prompts a crucial question: under what conditions do these

architectures become efficient? By delving into arithmetic expression evaluation and introducing the locality assumption, we identify scenarios where efficient Transformers can be efficient. Our theoretical results find empirical support through extensive experiments on tasks like Arithmetic, Longest Increasing Subsequence (LIS), and Edit Distance (ED). The observed dependency between hidden dimension and problem scale for efficient Transformers, in contrast to the stability in standard Transformers, validates our theory.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

Acknowledgement

Liwei Wang is supported by National Science and Technology Major Project (2022ZD0114902) and National Science Foundation of China (NSFC62276005). Di He is supported by National Science Foundation of China (NSFC62376007).

References

Akyürek, E., Schuurmans, D., Andreas, J., Ma, T., and Zhou, D. What learning algorithm is in-context learning? investigations with linear models. *arXiv preprint arXiv:2211.15661*, 2022.

Alberti, S., Dern, N., Thesing, L., and Kutyniok, G. Sumformer: Universal approximation for efficient transformers. In *Topological, Algebraic and Geometric Learning Workshops 2023*, pp. 72–86. PMLR, 2023.

Alman, J. and Song, Z. Fast attention requires bounded entries. *arXiv preprint arXiv:2302.13214*, 2023.

Beltagy, I., Peters, M. E., and Cohan, A. Longformer: The long-document transformer. *arXiv:2004.05150*, 2020.

Bhattamishra, S., Ahuja, K., and Goyal, N. On the ability and limitations of transformers to recognize formal languages. *arXiv preprint arXiv:2009.11264*, 2020.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. In *Advances in neural information processing systems*, volume 33, pp. 1877–1901, 2020.

Child, R., Gray, S., Radford, A., and Sutskever, I. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

Choromanski, K. M., Likhosherstov, V., Dohan, D., Song, X., Gane, A., Sarlos, T., Hawkins, P., Davis, J. Q., Mohiuddin, A., Kaiser, L., et al. Rethinking attention with performers. In *International Conference on Learning Representations*, 2021.

Dai, D., Sun, Y., Dong, L., Hao, Y., Ma, S., Sui, Z., and Wei, F. Why can gpt learn in-context? language models implicitly perform gradient descent as meta-optimizers. In *ICLR 2023 Workshop on Mathematical and Empirical Understanding of Foundation Models*, 2023.

Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q. V., and Salakhutdinov, R. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. 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)*, pp. 4171–4186. Association for Computational Linguistics, 2019.

Elhage, N., Nanda, N., Olsson, C., Henighan, T., Joseph, N., Mann, B., Askell, A., Bai, Y., Chen, A., Conerly, T., et al. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 1, 2021.

Feng, G., Gu, Y., Zhang, B., Ye, H., He, D., and Wang, L. Towards revealing the mystery behind chain of thought: a theoretical perspective. *Advances in Neural Information Processing Systems*, 2023.

Furst, M., Saxe, J. B., and Sipser, M. Parity, circuits, and the polynomial-time hierarchy. *Mathematical systems theory*, 17(1):13–27, 1984.

Garg, S., Tsipras, D., Liang, P., and Valiant, G. What can transformers learn in-context? a case study of simple function classes. In *Advances in Neural Information Processing Systems*, 2022.

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

Hahn, M. Theoretical limitations of self-attention in neural sequence models. *Transactions of the Association for Computational Linguistics*, 8:156–171, 2020.

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

Hendrycks, D. and Gimpel, K. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

Katharopoulos, A., Vyas, A., Pappas, N., and Fleuret, F. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International conference on machine learning*, pp. 5156–5165. PMLR, 2020.

Keles, F. D., Wijewardena, P. M., and Hegde, C. On the computational complexity of self-attention. In *International Conference on Algorithmic Learning Theory*, pp. 597–619. PMLR, 2023.

Kitaev, N., Kaiser, Ł., and Levskaya, A. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020.

Kojima, T., Gu, S. S., Reid, M., Matsuo, Y., and Iwasawa, Y. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems*, 2022.

Liu, B., Ash, J. T., Goel, S., Krishnamurthy, A., and Zhang, C. Transformers learn shortcuts to automata. *arXiv preprint arXiv:2210.10749*, 2022.

Liu, B., Ash, J. T., Goel, S., Krishnamurthy, A., and Zhang, C. Transformers learn shortcuts to automata. In *The Eleventh International Conference on Learning Representations*, 2023.

Loshchilov, I. and Hutter, F. Fixing weight decay regularization in adam. 2017.

Luo, S., Li, S., Cai, T., He, D., Peng, D., Zheng, S., Ke, G., Wang, L., and Liu, T.-Y. Stable, fast and accurate: Kernelized attention with relative positional encoding. In *Advances in Neural Information Processing Systems*, volume 34, pp. 22795–22807, 2021.

Merrill, W. and Sabharwal, A. The parallelism tradeoff: Limitations of log-precision transformers. *Transactions of the Association for Computational Linguistics*, 2023.

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

Nye, M., Andreassen, A. J., Gur-Ari, G., Michalewski, H., Austin, J., Bieber, D., Dohan, D., Lewkowycz, A., Bosma, M., Luan, D., Sutton, C., and Odena, A. Show your work: Scratchpads for intermediate computation with language models. In *Deep Learning for Code Workshop*, 2022.

Olsson, C., Elhage, N., Nanda, N., Joseph, N., DasSarma, N., Henighan, T., Mann, B., Askell, A., Bai, Y., Chen, A., Conerly, T., Drain, D., Ganguli, D., Hatfield-Dodds, Z., Hernandez, D., Johnston, S., Jones, A., Kernion, J., Lovitt, L., Ndousse, K., Amodei, D., Brown, T., Clark, J., Kaplan, J., McCandlish, S., and Olah, C. In-context learning and induction heads. *Transformer Circuits Thread*, 2022. https://transformer-circuits.pub/2022/in-context-learning-and-induction-heads/index.html.

OpenAI. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Peng, H., Pappas, N., Yogatama, D., Schwartz, R., Smith, N. A., and Kong, L. Random feature attention. *arXiv preprint arXiv:2103.02143*, 2021.

Pérez, J., Marinković, J., and Barceló, P. On the turing completeness of modern neural network architectures. *arXiv preprint arXiv:1901.03429*, 2019.

Pérez, J., Barceló, P., and Marinkovic, J. Attention is turing complete. *The Journal of Machine Learning Research*, 22(1):3463–3497, 2021.

Qiu, J., Ma, H., Levy, O., Yih, W.-t., Wang, S., and Tang, J. Blockwise self-attention for long document understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 2555–2565, 2020.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Roy, A., Saffar, M., Vaswani, A., and Grangier, D. Efficient content-based sparse attention with routing transformers. *Transactions of the Association for Computational Linguistics*, 9:53–68, 2021.

Sanford, C., Hsu, D., and Telgarsky, M. Representational strengths and limitations of transformers. *arXiv preprint arXiv:2306.02896*, 2023.

Sun, Y., Dong, L., Huang, S., Ma, S., Xia, Y., Xue, J., Wang, J., and Wei, F. Retentive network: A successor to transformer for large language models (2023). *URL http://arxiv. org/abs/2307.08621 v1*.

Tay, Y., Bahri, D., Yang, L., Metzler, D., and Juan, D.-C. Sparse sinkhorn attention. In *International Conference on Machine Learning*, pp. 9438–9447. PMLR, 2020a.

Tay, Y., Dehghani, M., Abnar, S., Shen, Y., Bahri, D., Pham, P., Rao, J., Yang, L., Ruder, S., and Metzler, D. Long range arena: A benchmark for efficient transformers. *arXiv preprint arXiv:2011.04006*, 2020b.

Tay, Y., Dehghani, M., Bahri, D., and Metzler, D. Efficient transformers: A survey, 2022.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P.,

Bhosale, S., et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Von Oswald, J., Niklasson, E., Randazzo, E., Sacramento, J., Mordvintsev, A., Zhmoginov, A., and Vladymyrov, M. Transformers learn in-context by gradient descent. In *International Conference on Machine Learning*, pp. 35151–35174. PMLR, 2023.

Vyas, A., Katharopoulos, A., and Fleuret, F. Fast transformers with clustered attention. In *Advances in Neural Information Processing Systems*, volume 33, pp. 21665–21674, 2020.

Wang, S., Li, B. Z., Khabsa, M., Fang, H., and Ma, H. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

Wei, C., Chen, Y., and Ma, T. Statistically meaningful approximation: a case study on approximating turing machines with transformers. *Advances in Neural Information Processing Systems*, 35:12071–12083, 2022a.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q. V., Zhou, D., et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35: 24824–24837, 2022b.

Weiss, G., Goldberg, Y., and Yahav, E. Thinking like transformers. In *International Conference on Machine Learning*, pp. 11080–11090. PMLR, 2021.

Yao, S., Peng, B., Papadimitriou, C., and Narasimhan, K. Self-attention networks can process bounded hierarchical languages. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 3770–3785, 2021.

Yun, C., Bhojanapalli, S., Rawat, A. S., Reddi, S. J., and Kumar, S. Are transformers universal approximators of sequence-to-sequence functions? *arXiv preprint arXiv:1912.10077*, 2019.

Zhou, D., Schärli, N., Hou, L., Wei, J., Scales, N., Wang, X., Schuurmans, D., Cui, C., Bousquet, O., Le, Q. V., and Chi, E. H. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations*, 2023.

A. Technical Lemmas

In this section, we propose some lemmas about the expressive power of the MLP, the linear transformer, and the sparse transformer. Due to the similarity between the efficient and the standard Transformer, we base some ideas and constructions on previous work (Feng et al., 2023).

A.1. Lemmas for MLP

In the previous work (Feng et al., 2023), the authors investigated the expressive power of the MLP with GeLU activation function. They showed that a two-layer MLP with GeLU activation and a fixed number of log-precision neurons can perform various tasks such as multiplication, linear transformation, and selection. Building on their work, our proof is more concise and clear. We will restate the relevant lemmas below, and refer to the appendices of Feng et al. (2023) for the proofs.

Lemma A.1 (From Feng et al. (2023))**.** *Let $f : \mathbb{R}^2 \to \mathbb{R}$ be a two-layer MLP with GeLU activation, and the hidden dimension is 4. Then, for any $\epsilon > 0$ and $M > 0$, there exist MLP parameters with ℓ_∞ norm upper bounded by $O(\mathrm{poly}(M, 1/\epsilon))$ such that $|f(a, b) - ab| \leq \epsilon$ holds for all $a, b \in [-M, M]$.*

Lemma A.2 (From Feng et al. (2023))**.** *Let $\boldsymbol{g} : \mathbb{R}^{d_1} \to \mathbb{R}^{d_2}$ be a two-layer MLP with* ReLU *activation, and all parameter values are upper bounded by M. Then, for any $\epsilon > 0$, there exists a two-layer MLP $\boldsymbol{f}$ of the same size with* GeLU *activation and parameters upper bounded by $O(\mathrm{poly}(M, 1/\epsilon))$ in the ℓ_∞ norm, such that for all $\boldsymbol{x} \in \mathbb{R}^{d_1}$, we have $\|\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{g}(\boldsymbol{x})\|_\infty \leq \epsilon$.*

Lemma A.3 (From Feng et al. (2023))**.** *Let $\boldsymbol{f} : \mathbb{R}^{d_1} \to \mathbb{R}^{d_2}$ be a two-layer MLP with* GeLU *activation, and the hidden dimension is $2d_2$. Let $\boldsymbol{W} \in \mathbb{R}^{d_2 \times d_1}$ be any matrix and denote $M = \max_{ij} |W_{ij}|$. Then, for any $\epsilon > 0$, there exist MLP parameters with ℓ_∞ norm bounded by $O(\mathrm{poly}(M, 1/\epsilon))$, such that for any $\boldsymbol{x} \in \mathbb{R}^{d_1}$, we have $\|\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{W}\boldsymbol{x}\|_\infty \leq \epsilon$.*

Lemma A.4 (From Feng et al. (2023))**.** *Define the selection function $\boldsymbol{g} : \mathbb{R}^d \times \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$ as follows:*

$$\boldsymbol{g}(\boldsymbol{x}, \boldsymbol{y}, t) = \begin{cases} \boldsymbol{x} & \text{if } t \geq 0, \\ \boldsymbol{y} & \text{if } t < 0. \end{cases} \tag{11}$$

Let $\boldsymbol{f} : \mathbb{R}^d \times \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$ be a two-layer MLP with GeLU activation, and the hidden dimension is $2d + 2$. Then, for any $\epsilon > 0$, $\alpha > 0$, and $M > 0$, there exist MLP parameters with ℓ_∞ norm bounded by $O(\mathrm{poly}(M, 1/\alpha, 1/\epsilon))$, such that for all $\boldsymbol{x} \in [-M, M]^d$, $\boldsymbol{y} \in [-M, M]^d$, and $t \in [-\infty, -\alpha] \cup [\alpha, +\infty]$, we have $\|\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y}, t) - \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{y}, t)\|_\infty \leq \epsilon$.

Lemma A.5. *For integer i, define $S_i = \left[i - \frac{1}{3}, i + \frac{1}{3}\right]$. Let $n \in \mathbb{Z}$ and $D = \cup_{i=0}^{n} S_i$. Define the function $\boldsymbol{g} : D \to \mathbb{R}^n$ as follows: $\boldsymbol{f}(x) = \boldsymbol{m}_i$ if $x \in S_i$, where $\boldsymbol{m}_i$ has its first i entries 1 and the rest entries 0. Let $\boldsymbol{f} : \mathbb{R} \to \mathbb{R}^n$ be a two-layer MLP with GeLU activation, and the hidden dimension is $2n$. Then, for any $\epsilon > 0$, $\alpha > 0$, there exists MLP parameters with l_∞ norm bounded by $O(\mathrm{poly}(n, 1/\epsilon))$, such that for all $x \in D$, we have $\|\boldsymbol{f}(x) - \boldsymbol{g}(x)\|_\infty \leq \epsilon$.*

Proof. Notice that the k-th entry of $\boldsymbol{m}_j$ is

$$\mathbb{I}[k \leq j] = \mathrm{ReLU}\left[-2k + 2\left(j + \frac{3}{4}\right)\right] - \mathrm{ReLU}\left[-2k + 2\left(j + \frac{1}{4}\right)\right] \tag{12}$$

Thus each entry of $\boldsymbol{m}_j$ can be implemented by an MLP with hidden dimension 2. By Lemma A.2, we can perform both tasks using an MLP with GeLU activation, with hidden dimension $2n$. □

Lemma A.6. *For integer i, define $S_i = \left[i - \frac{1}{3}, i + \frac{1}{3}\right]$. Let $n \in \mathbb{Z}$ and $D = \cup_{i=0}^{n} S_i$. Define the function $\boldsymbol{g} : D \to \mathbb{R}^n$ as follows: $\boldsymbol{f}(x) = \boldsymbol{n}_i$ if $x \in S_i$, where $\boldsymbol{n}_i$ has its first $i - 1$ entries 0 and the rest entries 1. Let $\boldsymbol{f} : \mathbb{R} \to \mathbb{R}^n$ be a two-layer MLP with GeLU activation, and the hidden dimension is $2n$. Then, for any $\epsilon > 0$, $\alpha > 0$, there exists MLP parameters with l_∞ norm bounded by $O(\mathrm{poly}(n, 1/\epsilon))$, such that for all $x \in D$, we have $\|\boldsymbol{f}(x) - \boldsymbol{g}(x)\|_\infty \leq \epsilon$.*

Proof. Similarly to the previous proof, the k-th entry of $\boldsymbol{n}_j$ is

$$\mathbb{I}[k \geq j] = \mathrm{ReLU}\left[2k - 2\left(j - \frac{3}{4}\right)\right] - \mathrm{ReLU}\left[2k - 2\left(j - \frac{1}{4}\right)\right] \tag{13}$$

which indicates that each entry of $\boldsymbol{n}_j$ can be implemented by an MLP with hidden dimension 2. By Lemma A.2, we can perform both tasks using an MLP with GeLU activation, with hidden dimension $2n$. □

Lemma A.7. *For integer i, define $S_i = \left[i - \frac{1}{4}, i + \frac{1}{4}\right]$. Given positive integer n and integer $i \in [1, n^2]$. There exists a two-layer MLP $\boldsymbol{f} : \mathbb{R} \to \mathbb{R}^3$ with GeLU activation and hidden dimension $O(n)$, such that for any integer $i \in [1, n^2]$ and $x \in S_i$, $\boldsymbol{f}(x) = (\lceil i/n \rceil, \lceil i/n \rceil - 1, n - (\lceil i/n \rceil \times n - i))$. Here, $n - (\lceil i/n \rceil \times n - i) = i$ mod n if i mod $n \neq 0$, and $n - (\lceil i/n \rceil \times n - i) = n$ if i mod $n = 0$.*

Proof. Since $i \in \{1, 2, \cdots, n^2\}$, we can get $\lceil i/n \rceil \in \{1, 2, \cdots, n\}$. By the proposition that i is an integer, we can use an MLP with hidden dimension $2n$ and ReLU activation to calcualte $\lceil i/n \rceil$ as following:

$$\lceil i/n \rceil = \sum_{j=1}^{n} \mathbb{I}[i \leq jn] = \sum_{j=1}^{n} \mathrm{ReLU}\left[-2i + 2\left(jn + \frac{3}{4}\right)\right] - \mathrm{ReLU}\left[-2i + 2\left(jn + \frac{1}{4}\right)\right] \tag{14}$$

Thus we can calculate $\lceil i/n \rceil$, $\lceil i/n \rceil - 1$, $n - (\lceil i/n \rceil \times n - i)$ task by an MLP with ReLU activation with hidden dimension $O(n)$. By Lemma A.2, we can finish these tasks by an MLP with GeLU activation with hidden dimension $O(n)$. □

A.2. Lemmas for Linear Transformer

Lemma A.8. *The attention module in Linear Transformer*

$$\mathrm{Attn}(\boldsymbol{x}_i) = \frac{\sum_{j \leq i} \phi(\boldsymbol{W}_K \boldsymbol{x}_j)^\top \phi(\boldsymbol{W}_Q \boldsymbol{x}_i) \boldsymbol{W}_V \boldsymbol{x}_i}{\sum_{j \leq i} \phi(\boldsymbol{W}_K \boldsymbol{x}_j)^\top \phi(\boldsymbol{W}_Q \boldsymbol{x}_i)} \tag{15}$$

can be implemented as follows:

$$\boldsymbol{s}_0 = \boldsymbol{0},\ \boldsymbol{z}_0 = \boldsymbol{0} \tag{16}$$

$$\boldsymbol{s}_i = \boldsymbol{s}_{i-1} + \phi(\boldsymbol{W}_K \boldsymbol{x}_i)(\boldsymbol{W}_V \boldsymbol{x}_i)^\top \tag{17}$$

$$\boldsymbol{z}_i = \boldsymbol{z}_{i-1} + \phi(\boldsymbol{W}_K \boldsymbol{x}_i) \tag{18}$$

$$\mathrm{Attn}(\boldsymbol{x}_i)^\top = \frac{\phi(\boldsymbol{W}_Q \boldsymbol{x}_i)^\top \boldsymbol{s}_i}{\phi(\boldsymbol{W}_Q \boldsymbol{x}_i)^\top \boldsymbol{z}_i} \tag{19}$$

Since this implementation is quite similar to RNNs, we called $\boldsymbol{s}, \boldsymbol{z}$ in our implementation hidden states.

Now, we introduce the AGG operation for Linear Transformer. Let m be an integer and let $\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_{m^2}$ be a sequence of vectors where $\boldsymbol{x}_i = (\hat{\boldsymbol{x}}_i, \boldsymbol{e}_{\lceil i/m \rceil}, \boldsymbol{e}_{q_i}, 1) \in [-M, M]^{d+m+1}$, $\hat{\boldsymbol{x}}_i \in \mathbb{R}^d$, and $\boldsymbol{e}_{\lceil i/m \rceil}, \boldsymbol{e}_{q_i}$ are one-hot vectors in $\mathbb{R}^m$, and M is a large constant. Let $\mathcal{S}_i = \{j \in \mathbb{Z} : q_i m - (m-1) \leq j \leq \min(q_i m, i)\}$. Define the AGG operation as follows: The output is a sequence of vectors $\boldsymbol{u}_1, \cdots, \boldsymbol{u}_{m^2}$ with $\boldsymbol{u}_i = \mathrm{mean}_{j \in \mathcal{S}_i} \hat{\boldsymbol{x}}_j$. The output is undefined when $\mathcal{S}_i = \varnothing$.

Lemma A.9. *For any $0 < \epsilon \leq M$, there exists an linear attention layer with hidden dimension $O(\max(d, m))$ and one causal attention head that can approximate the AGG operation defined above. Specifically, suppose the attention output are $\boldsymbol{o}_1, \cdots, \boldsymbol{o}_{m^2}$, then we have $\|\boldsymbol{o}_i - \boldsymbol{u}_i\|_\infty \leq \epsilon$ for all $i \in [m^2]$. Moreover, the l_∞ norm of attention parameters are bounded by $O(\mathrm{poly}(\log(M), \log(m), \log(1/\epsilon))$.*

Proof. Without loss of generosity, we can assume $d = m$ (otherwise, we can do some padding with 0). We will provide a proof based on the implement of Lemma A.8.

We construct the query, key and value vectors as follows:

- Query: $\boldsymbol{q}_i = \mu \boldsymbol{e}_{q_i} - \mu$
- Key: $\boldsymbol{k}_i = \mu \boldsymbol{e}_{\lceil i/m \rceil} - \mu$
- Value: $\boldsymbol{v}_i = \hat{\boldsymbol{x}}_i$

where $\mu > 0$ is a constant defined later. This can be achieved by setting appropriate $\boldsymbol{W}_Q, \boldsymbol{W}_K, \boldsymbol{W}_V$.

Notice that $j \in \mathcal{S}_i$ if and only if $j \leq i$ and $\lceil j/m \rceil = q_i$. By the construction of $\boldsymbol{q}_i, \boldsymbol{k}_i, \boldsymbol{v}_i$, we can get

$$a_{ij} := \boldsymbol{q}_i^\top \boldsymbol{k}_j = \begin{cases} 1 + (m-1)\exp(-\mu), & j \in \mathcal{S}_i \\ 2\exp(-\mu) + (m-2)\exp(-2\mu), & j \notin \mathcal{S}_i \end{cases}$$

By taking $\mu = \ln \frac{2Mm^3}{\epsilon}$, which is bounded by $O(\text{poly}(\log(M), \log(m), \log(1/\epsilon))$, we have

$$\begin{aligned}
\sum_{j \notin \mathcal{S}_i} a_{ij} &\leq \frac{(m^2 - |\mathcal{S}_i|)[2\exp(-\mu) + (m-2)\exp(-2\mu)]}{(m^2 - |\mathcal{S}_i|)[2\exp(-\mu) + (m-2)\exp(-2\mu)] + |\mathcal{S}_i|[1 + (m-1)\exp(-\mu)]} \\
&= \frac{1}{1 + \frac{|\mathcal{S}_i|}{m^2 - |\mathcal{S}_i|} \cdot \frac{1 + (m-1)\exp(-\mu)}{2\exp(-\mu) + (m-2)\exp(-2\mu)}} \\
&\leq \frac{m^2 - |\mathcal{S}_i|}{|\mathcal{S}_i|} \cdot \frac{2\exp(-\mu) + (m-2)\exp(-2\mu)}{1 + (m-1)\exp(-\mu)} \\
&\leq m^2 \cdot m\exp(-\mu) = m^3 \exp(-\mu) = \frac{\epsilon}{2M}
\end{aligned}$$

Similarly, for $j \in \mathcal{S}_i$, we have

$$\begin{aligned}
&\left| a_{ij} - \frac{1}{|\mathcal{S}_i|} \right| \\
&\leq \frac{1}{|\mathcal{S}_i|} - \frac{1 + (m-1)\exp(-\mu)}{(m^2 - |\mathcal{S}_i|)[2\exp(-\mu) + (m-2)\exp(-2\mu)] + |\mathcal{S}_i|[1 + (m-1)\exp(-\mu)]} \\
&= \frac{1}{|\mathcal{S}_i|} - \frac{1}{(m^2 - |\mathcal{S}_i|)\frac{2\exp(-\mu) + (m-2)\exp(-2\mu)}{1 + (m-1)\exp(-\mu)} + |\mathcal{S}_i|} \\
&= \frac{(m^2 - |\mathcal{S}_i|)\frac{2\exp(-\mu) + (m-2)\exp(-2\mu)}{1 + (m-1)\exp(-\mu)}}{|\mathcal{S}_i| \left[(m^2 - |\mathcal{S}_i|)\frac{2\exp(-\mu) + (m-2)\exp(-2\mu)}{1 + (m-1)\exp(-\mu)} + |\mathcal{S}_i| \right]} \\
&\leq \frac{1}{|\mathcal{S}_i|}(m^2 - |\mathcal{S}_i|) \cdot [2\exp(-\mu) + (m-2)\exp(-2\mu)] \\
&\leq \frac{1}{|\mathcal{S}_i|} m^2 \cdot m\exp(-\mu) = \frac{1}{|\mathcal{S}_i|} m^3 \exp(-\mu) = \frac{\epsilon}{2M|\mathcal{S}_i|}
\end{aligned}$$

We thus obtain

$$\|\boldsymbol{o}_i - \boldsymbol{u}_i\|_\infty = \left\| \sum_j a_{ij}\hat{\boldsymbol{x}}_j - \frac{1}{|\mathcal{S}_i|} \sum_{j \in \mathcal{S}_i} \hat{\boldsymbol{x}}_j \right\|_\infty \leq \max_j \|\hat{\boldsymbol{x}}_j\|_\infty \cdot \left(\sum_{j \notin \mathcal{S}_i} a_{ij} + \sum_{j \in \mathcal{S}_i} \left| a_{ij} - \frac{1}{|\mathcal{S}_i|} \right| \right) \leq \epsilon$$

which concludes our proof. □

A.3. Lemmas for Sparse Transformer

The previous work by Feng et al. (2023) studied the expressive power of the standard attention layer and introduced a basic operation, COPY, that can be implemented by the attention layer. In this section, we introduce an updated version of the COPY operation for the sparse transformer and demonstrate that the sparse transformer module is capable of implementing this operation.

Firstly, we will enumerate the elements and their respective notations that will be employed to define the COPY operation.

- A sequence of vectors $\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n \in \mathbb{R}^{d+2}$, where $\boldsymbol{x}_i = [\boldsymbol{v}_i, r_i, i \bmod B, \lceil \frac{i}{B} \rceil, 1]$, $\boldsymbol{x}_i \in \mathbb{R}^d$, $r_i \in \mathbb{R}$, $B \in \mathbb{Z}$.
- Three matrices $\boldsymbol{K}, \boldsymbol{Q}, \boldsymbol{V} \in \mathbb{R}^{d' \times (d+2)}$, where $\|V\|_\infty \leq 1$
- The attention sets $\mathcal{S}'_i$ for each i, where $\mathcal{S}'_i$ is the indices of the tokens attended by the i-th token.
- Two real number $\rho, \delta > 0$

Denoting that $\boldsymbol{k}_i = \boldsymbol{K}\boldsymbol{x}_i, \boldsymbol{q}_i = \boldsymbol{Q}\boldsymbol{x}_i, \boldsymbol{v}_i = \boldsymbol{V}\boldsymbol{x}_i \in \mathbb{R}^{d'}$, we can define the matching set $\mathcal{S}_i$ for the i as $\mathcal{S}_i = \{j < i \mid \boldsymbol{q}_i \cdot \boldsymbol{k}_j < \rho\} \cap \mathcal{S}'_i$. Then the output of the COPY operation $\boldsymbol{u}_1, \boldsymbol{u}_2, \cdots, \boldsymbol{u}_n$ is defined as $\boldsymbol{u}_i = \boldsymbol{v}_{\text{pos(i)}}$, where $\text{pos}(i) = \text{argmax}_{j \in \mathcal{S}_i} r_j$.

Intuitively, the COPY operation copies the embedding of the matching token with the highest rating. Note that the output of the COPY operation can be anything when $\mathcal{S}_i$ is empty.

Moreover, we make the following regularity assumption:

Assumption A.10. The input sequence $\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n$ and the matrices $\boldsymbol{K}, \boldsymbol{Q}, \boldsymbol{V}$ satisfy the following condition:

- For any $i, j \in [n]$, either $|q_i \cdot k_j| \leq \rho$ or $q_i \cdot k_j \leq -\delta$
- For any $i, j \in [n]$, either $i = j$ or $|r_i - r_j| \geq \delta$
- $\|\boldsymbol{V}\|_\infty \leq 1$

Assumption A.10 ensures that only one token with the highest rating is matched, and there is a substantial difference between this token and others. The subsequent lemma demonstrates that the sparse transformer module can effectively implement the COPY operation.

Lemma A.11 (COPY Operation of Standard Transformer, From Feng et al. (2023))**.** *Given that Assumption A.10 holds with $\rho \leq \frac{\delta^2}{8M}$, there exists a standard transformer module defined in Equation (6) with one attention layer, block size B, embedding size $O(d)$, and one attention head that can approximate the COPY operation defined above. Specifically, for any sequence of vectors $\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n$, denote the corresponding output of the attention layer as $\boldsymbol{o}_1, \boldsymbol{o}_2, \cdots, \boldsymbol{o}_n$. Then, we have $\|o_i - u_i\|_\infty \leq \epsilon$ for all $i \in [n]$ with $\mathcal{S}_i \neq \emptyset$. Furthermore, the ℓ_∞ norm of attention parameters is bounded by $O(\text{poly}(M, 1/\delta, \log(n), \log(1/\epsilon)))$.*

Now, we will prove a special form of the COPY operation for the Sparse Transformer, such that given an index of a token, COPY the embedding of the token with the given index.

Lemma A.12 (COPY Operation of Sparse Transformer)**.** *Given that Assumption A.10 holds with $\rho \leq \frac{\delta^2}{8M}$, there exists a sparse transformer module defined in Equation (6) with three attention layer, block size B, embedding size $O(d)$, and one attention head that can approximate the COPY operation defined above. Specifically, for any sequence of vectors $\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n$, denote the corresponding output of the attention layer as $\boldsymbol{o}_1, \boldsymbol{o}_2, \cdots, \boldsymbol{o}_n$. Then, we have $\|o_i - u_i\|_\infty \leq \epsilon$ for all $i \in [n]$ with $\mathcal{S}_i \neq \emptyset$. Furthermore, the ℓ_∞ norm of attention parameters is bounded by $O(\text{poly}(M, 1/\delta, \log(n), \log(1/\epsilon)))$.*

The proof for this version is an extension of the proof for a Lemma described in Feng et al. (2023). For the reader's convenience, we provide a concise proof as follows.

Proof. **Layer 1.** In our formulation, the input embedding $\boldsymbol{x}_i$ has the form $\boldsymbol{x}_i = (\boldsymbol{v}_i, p_{\text{target}}, i \bmod B, \lceil \frac{i}{B} \rceil, 1)$, where p_{target} is the target index of the COPY operation. The primary objective of the first layer is to utilize its MLP to transform the input embedding. The target output of this layer is

$$[\boldsymbol{v}_{i,1}, \boldsymbol{v}_{i,2}, \cdots, \boldsymbol{v}_{i,B}, p_{\text{target}}, i \bmod B, \lceil \frac{i}{B} \rceil, 1], \quad \text{where} \quad \boldsymbol{v}_{i,j} = \begin{cases} \mathbf{0} & \text{, when } j \neq i \bmod B + 1 \\ \boldsymbol{v}_i & \text{, when } j = i \bmod B + 1 \end{cases}$$

Note that v_i is bound by $[-M, M]$. Therefore, we can express $\boldsymbol{v}_{i,j}$ as

$$\boldsymbol{v}_{i,j} = \text{ReLU}\Big(\text{ReLU}(\boldsymbol{v}_i + M) - M \cdot \big(1 + \cdot\text{ReLU}(j - i \bmod B) + \cdot\text{ReLU}(i \bmod B - j)\big)\Big)$$

According to Lemma A.2, an MLP with GeLU as the activation function can accomplish this task with arbitrarily small error. Therefore, we can write the output of this layer as $\boldsymbol{x}_i^{(1)} = [\boldsymbol{v}_{i,1}, \boldsymbol{v}_{i,2}, \cdots, \boldsymbol{v}_{i,B}, p_{\text{target}}, i \bmod B, \lceil \frac{i}{B} \rceil, 1]$.

Layer 2. The main task of the second layer is to sum over the embedding of previous B tokens. In this layer, the attention head just pays uniform attention to the previous B tokens. Note that, in the first layer, we distribute the embeddings of different tokens on the different dimensions of the embedding. Therefore, the output of the attention layer is $\frac{1}{B} \cdot [\boldsymbol{v}_{(i-B+i \bmod B+1)}, \boldsymbol{v}_{(i-B+i \bmod B+2)}, \cdots, \boldsymbol{v}_i, \cdots \boldsymbol{v}_{(i-B+i \bmod B)}, p_{\text{target}}, i \bmod B, \lceil \frac{i}{B} \rceil, 1]$. Then, by using the MLP to multiply the number of tokens B, we can get the embeddings of all tokens. Moreover, in this layer, according to Lemma A.7 we can use the MLP to calculate the index of the block of the target token as $\lceil \frac{p_{\text{target}}}{B} \rceil$. The output of this layer as

$$\boldsymbol{x}_i^{(2)} = [\boldsymbol{v}_{(i-B+i \bmod B+1)}, \boldsymbol{v}_{(i-B+i \bmod B+2)}, \cdots, \boldsymbol{v}_i, \cdots \boldsymbol{v}_{(i-B+i \bmod B)}, p_{\text{target}}, i \bmod B, \lceil \frac{i}{B} \rceil, \lceil \frac{p_{\text{target}}}{B} \rceil, 1].$$

Layer 3. The main task of the second layer is to COPY the embedding of the token in the corresponding block and select the target embedding. In this layer, the attention head just COPY the embedding of the token in the corresponding block. According to Lemma A.11, the attention layer can complete this task with arbitrarily small errors. Finally, we use the MLP to select the target embedding from the copied embedding and obtain the final output. □

B. Proofs of the theorems in Section 4

For ease of reading, we reclaim some important assumptions here, which are natural and mild.

Assumption B.1. Each function f, $\boldsymbol{g}$, $\boldsymbol{h}$ and u defined in CoT for DP can be approximated by constant size MLP with GeLU activation.

Assumption B.2. The state transition function F, where the inputs are the token number and current state, and the outputs are the next state, can be approximated with MLP and GeLU function.

In addition we replace all residual connections in attention layers and MLP with concatenation, which doesn't change the expressive power of model architecture.

B.1. Proof of the Theorem 4.2

In this subsection, we provide a complete proof of Theorem 4.2. For clarity, we restate the theorem for the linear transformer and sparse transformer individually and provide their respective proofs.

Theorem B.3. *Consider any DP problem satisfying the same condition as in Theorem 4.1. Given any integer $n > 0$, let L be the length of the output sequence when the input sequence length is n. Then, there is a (log-precision) linear transformer with a constant depth M, a constant number of attention heads H, and a hidden dimension $D = \mathrm{O}(\sqrt{L})$ that can generate the correct output for all inputs $\boldsymbol{s}$ of length n.*

Proof. Denote $W = \lceil \sqrt{L} \rceil$. In this formulation, the embeddings are $\boldsymbol{x}_k^{(0)} = (\boldsymbol{e}_k^{\text{input}}, \boldsymbol{e}_k^{\text{state}}, \boldsymbol{e}_k^{\text{dp}}, \boldsymbol{e}_k^{\text{answer}}, k, 1)$, where $\boldsymbol{e}_k^{\text{input}}, \boldsymbol{e}_k^{\text{state}}, \boldsymbol{e}_k^{\text{dp}}, \boldsymbol{e}_k^{\text{answer}}$ corresponds to input token, DP state, DP value, final result. The embeddings are $\mathbf{0}$ if it's not defined. We construct the layers as follows.

Block 1. According to Assumption B.2, we can use constant layers of MLPs to obtain $\boldsymbol{e}_k^{\text{next_state}}$ with input $\boldsymbol{e}_k^{\text{state}}$. This can be done by setting zero weight matrices in attention layers and discard attention output in linear projection of MLP. The output of this block is $\boldsymbol{x}_k^{(1)} = (\boldsymbol{e}_k^{\text{input}}, \boldsymbol{e}_k^{\text{state}}, \boldsymbol{e}_k^{\text{next_state}}, \boldsymbol{e}_k^{\text{dp}}, k, 1)$.

Block 2. The second layer of the Transformer finishes the following tasks:

- Calculate $\boldsymbol{h}(\boldsymbol{e}_k^{\text{next_state}}), \boldsymbol{g}(\boldsymbol{e}_k^{\text{next_state}})$.
- Calculate f_k^{state} as the indicator variable of $\boldsymbol{e}_k^{\text{state}}$ is the last state.
- Calculate $(\boldsymbol{h}(\boldsymbol{e}_k^{\text{next_state}}))^2, (\boldsymbol{g}(\boldsymbol{e}_k^{\text{next_state}}))^2, (\boldsymbol{e}_k^{\text{state}})^2$, which are element-wise square operations, and k^2.

The first task can be implemented by several layers of MLPs by Assumption B.1. To perform the second task, we can check whether $\boldsymbol{e}_k^{\text{state}} \neq \mathbf{0}$ and $\boldsymbol{e}_k^{\text{next_state}} = \mathbf{0}$. The last task can be done with an MLP using Lemma A.1. The output of this block is

$$\begin{aligned}\boldsymbol{x}_k^{(2)} = (&\boldsymbol{e}_k^{\text{input}}, \boldsymbol{e}_k^{\text{state}}, \boldsymbol{e}_k^{\text{next_state}}, \boldsymbol{e}_k^{\text{dp}}, \boldsymbol{e}_k^{\text{sep}}, \boldsymbol{h}(\boldsymbol{e}_k^{\text{next_state}}), \boldsymbol{g}(\boldsymbol{e}_k^{\text{next_state}}),\\ &(\boldsymbol{h}(\boldsymbol{e}_k^{\text{next_state}}))^2, (\boldsymbol{g}(\boldsymbol{e}_k^{\text{next_state}}))^2, (\boldsymbol{e}_k^{\text{state}})^2, f_k^{\text{state}}, k, k^2, 1).\end{aligned}$$

Block 3. The third block of the Transformer uses $K + J$ heads to perform the following tasks where K and J are defined in (9), refers to the transition in DP:

- Get inputs of $s_{g_1(i)}, \cdots, s_{g_J(i)}$ where i corresponds to $\boldsymbol{e}_k^{\text{next_state}}$.
- Get DP value of $\mathsf{dp}(h_1(i)), \cdots, \mathsf{dp}(h_K(i))$ for i corresponds to $\boldsymbol{e}_k^{\text{next_state}}$.
- Calculate $\boldsymbol{e}_k^{\text{next_dp}}$, i.e., the DP value of the next state, $\boldsymbol{e}_k^{\text{next_state}}$.

To perform the first two tasks, we need to first calculate the absolute position of the embeddings we want. This can be implemented by several MLPs by Assumption B.1 and problem size n. Suppose the absolute positions are $\hat{g}_1, \cdots, \hat{g}_J, \hat{h}_1, \cdots, \hat{h}_K$. Then we can do the following tasks for $t = \hat{g}_1, \cdots, \hat{g}_J, \hat{h}_1, \cdots, \hat{h}_K$:

1. Calculate $\lceil t/W \rceil, W - (\lceil t/W \rceil \times W - t)$ by Lemma A.7.
2. Calculate $\boldsymbol{e}_s \in \mathbb{R}^W$ for $s = \lceil t/W \rceil, W - (\lceil t/W \rceil \times W - t)$ by Lemma A.5, A.6 and the fact that $\boldsymbol{e}_i = \boldsymbol{m}_i \odot \boldsymbol{n}_i$.

We also calculate $\boldsymbol{e}_s \in \mathbb{R}^W$ for $s = \lceil k/W \rceil, W - (\lceil k/W \rceil \times W - k)$ by Lemma A.5, A.6. Then we can get

$$(\mathbb{I}[W - (\lceil k/W \rceil \times W - k) = 1] \cdot \boldsymbol{x}_k^{(2)}, \cdots, \mathbb{I}[W - (\lceil k/W \rceil \times W - k) = 1] \cdot \boldsymbol{x}_k^{(2)})$$

by the multiplication between $\boldsymbol{e}_{W-(\lceil k/W \rceil \times W - k)}$ and $\boldsymbol{x}_k^{(2)}$ which can be implemented by an MLP with hidden dimension $O(W)$. Then, we can get $\boldsymbol{x}_t^{(2)}$ for $t = \hat{g}_1, \cdots, \hat{g}_J, \hat{h}_1, \cdots, \hat{h}_K$ in following steps:

1. Get
$$\begin{cases} \frac{1}{W}(\boldsymbol{x}^{(2)}_{\lceil t/W \rceil \times W - (W-1)}, \cdots, \boldsymbol{x}^{(2)}_{\lceil t/W \rceil \times W}), & i \geq \lceil t/W \rceil \times W \\ \frac{1}{i - \lceil t/W \rceil \times (W-1)}(\boldsymbol{x}^{(2)}_{\lceil t/W \rceil \times W - (W-1)}, \cdots, \boldsymbol{x}^{(2)}_i, \boldsymbol{0}, \cdots, \boldsymbol{0}), & i < \lceil t/W \rceil \times W \end{cases}$$
for $t = \hat{g}_1, \cdots, \hat{g}_J, \hat{h}_1, \cdots, \hat{h}_K$ by using Lemma A.9.
2. Get
$$\begin{cases} (\boldsymbol{x}^{(2)}_{\lceil t/W \rceil \times W - (W-1)}, \cdots, \boldsymbol{x}^{(2)}_{\lceil t/W \rceil \times W}), & i \geq \lceil t/W \rceil \times W \\ (\boldsymbol{x}^{(2)}_{\lceil t/W \rceil \times W - (W-1)}, \cdots, \boldsymbol{x}^{(2)}_i, \boldsymbol{0}, \cdots, \boldsymbol{0}), & i < \lceil t/W \rceil \times W \end{cases}$$
for $t = \hat{g}_1, \cdots, \hat{g}_J, \hat{h}_1, \cdots, \hat{h}_K$ by using Lemma A.1 for multiplication first, and use Lemma A.4 to implement a conditional selection.
3. Get $\boldsymbol{x}_t^{(2)}$ for $t = \hat{g}_1, \cdots, \hat{g}_J, \hat{h}_1, \cdots, \hat{h}_K$ by multiplication between the above vector and $\boldsymbol{e}_{W-(\lceil t/W \rceil \times W - t)}$, then sum up them. Then we can get $s_{g_1^{(\boldsymbol{n})}(i)}, \cdots, s_{g_J^{(\boldsymbol{n})}(i)}; \mathsf{dp}(h_1^{(\boldsymbol{n})}(i)), \cdots, \mathsf{dp}(h_K^{(\boldsymbol{n})}(i))$ by linear projection.

If the corresponding field is undefined, we can mark them as some special value. This can be done by selection using MLP (Lemma A.4). Assumption B.1 indicates that we can calculate the function f (defined in (9)) using an MLP. The output of this layer is

$$\boldsymbol{x}_k^{(3)} = (\boldsymbol{e}_k^{\text{next_state}}, \boldsymbol{e}_k^{\text{dp}}, \boldsymbol{e}_k^{\text{next_dp}}, \boldsymbol{n}_k, f_k^{\text{state}}, k, 1).$$

Block 4. The fourth block of the autoregressive transformer generates the output based on the flag f_k^{state}. We calculate $u(\boldsymbol{e}_k^{\text{dp}})$ to get the final result. If $f_k^{\text{state}} = 1$, then we select $u(\boldsymbol{e}_k^{\text{dp}})$ as the output; otherwise, we prepare and output the DP result for the next state, i.e., $\boldsymbol{e}_k^{\text{next_state}}$ and $\boldsymbol{e}_k^{\text{next_dp}}$. This is a conditional selection operation and thus can be implemented by an MLP (Lemma A.4). □

Theorem B.4. *Consider any DP problem satisfying the same condition as in Theorem 4.1. Given any integer $n > 0$, let L be the length of the output sequence when the input sequence length is n. Then, there is a (log-precision) sparse Transformer with block size $B = \Theta(\sqrt{L})$ with a constant depth M, a constant number of attention heads H, and a hidden dimension $D = \mathrm{O}(\sqrt{L})$ that can generate the correct output for all inputs $\boldsymbol{s}$ of length n.*

Proof. The construction of the first two blocks is the same as the linear transformer, and we will give the construction of the third block and the fourth block for the sparse transformer.

Block 3. The third block of the Transformer uses $K + J$ heads to perform the following tasks where K and J are defined in (9), refers to the transition in DP:

- Get inputs of $s_{g_1(i)}, \cdots, s_{g_J(i)}$ where i corresponds to $\boldsymbol{e}_k^{\text{next_state}}$.
- Get DP value of $\mathsf{dp}(h_1(i)), \cdots, \mathsf{dp}(h_K(i))$ for i corresponds to $\boldsymbol{e}_k^{\text{next_state}}$.
- Calculate $\boldsymbol{e}_k^{\text{next_dp}}$, i.e., the DP value of the next state, $\boldsymbol{e}_k^{\text{next_state}}$.

To perform the first two tasks, we need to first calculate the absolute position of the embeddings we want. This can be implemented by several MLPs by Assumption B.1 and problem size n. Supposing the absolute positions are $\hat{g}_1, \cdots, \hat{g}_J, \hat{h}_1, \cdots, \hat{h}_K$, we can perform the COPY operation by Lemma A.12, and we can obtain the input $s_{g_1(i)}, \cdots, s_{g_J(i)}$ and the DP value of $\mathsf{dp}(h_1(i)), \cdots, \mathsf{dp}(h_K(i))$ for i corresponds to $\boldsymbol{e}_k^{\text{next_state}}$. With the inputs and DP value, according to Assumption B.1, we can calculate the function f (defined in (9)) using an MLP. The output of this layer is

$$\boldsymbol{x}_k^{(3)} = (\boldsymbol{e}_k^{\text{next_state}}, \boldsymbol{e}_k^{\text{dp}}, \boldsymbol{e}_k^{\text{next_dp}}, \boldsymbol{n}_k, f_k^{\text{state}}, k, 1).$$

Block 4. The fourth block of the sparse transformer generates the output based on the flag f_k^{state}. We calculate $u(\boldsymbol{e}_k^{\text{dp}})$ to get the final result. If $f_k^{\text{state}} = 1$, then we select $u(\boldsymbol{e}_k^{\text{dp}})$ as the output; otherwise, we prepare and output the DP result for the next state, i.e., $\boldsymbol{e}_k^{\text{next_state}}$ and $\boldsymbol{e}_k^{\text{next_dp}}$. This is a conditional selection operation and thus can be implemented by an MLP (Lemma A.4). □

B.2. The proof of Theorem 4.4

In this subsection, we provide a complete proof of Theorem 4.2. For clarity, we restate the theorem for the linear transformer and sparse transformer individually and provide their respective full proofs.

Theorem B.5. *Consider any regular DP problem satisfying the same condition as in Theorem 4.1. Assume that the output sequence length L is proportional to the input sequence length n, i.e., $L = \Theta(n)$. Then, given a sufficiently large n, for (log-precision) linear Transformer, a model with a constant depth M and a constant number of attention heads H can generate the correct output for all inputs $\boldsymbol{s}$ of length n only if the hidden dimension $D = \tilde{\Omega}(\sqrt{L})$.*

Proof. By Lemma A.8, we can know the hidden states corresponding to each Linear Transformer layer and the last input token (which is a fixed special token) determines the CoT output. Thus, the size of hidden states should be at least $\Omega(n) = \Omega(L)$. By the regularity assumption, we know that different input should corresponds to different hidden states. This implies that the hidden dimension should be at least $\Omega(\sqrt{L/\log L}) = \tilde{\Omega}(\sqrt{L})$, concluding our proof. □

Theorem B.6. *Consider any regular DP problem satisfying the same condition as in Theorem 4.1. Assume that the output sequence length L is proportional to the input sequence length n, i.e., $L = \Theta(n)$. Then, given a sufficiently large n, for (log-precision) sparse Transformer with block size $B = \Theta(\sqrt{L})$, a model with a constant depth M and a constant number of attention heads H can generate the correct output for all inputs $\boldsymbol{s}$ of length n only if the hidden dimension $D = \tilde{\Omega}(\sqrt{L})$.*

Proof. Assuming $D = o(\frac{\sqrt{L}}{\ln L})$, we present a proof by contradiction. When the model generates the output sequence, the model attends to at most $\Theta(B)$ tokens in the input sequence, which are the last c tokens of every block and the last B tokens of the input sequence. The overall memory cost of all the hidden embeddings for these tokens is $\Theta(BD\log n)$ bits. According to Equation (6), given the same hidden embedding of these tokens in every layer, the model will execute the same computation and produce the identical sequence. Thus, the model can produce a maximum of $e^{\Theta(BD\log n)} = e^{o(L)}$ output sequence types given the hidden dimension $D = o(\frac{\sqrt{L}}{\ln L})$. However, for an input sequence of length L, there are corresponding $e^{\Theta(L)}$ input and output sequence types. According to the pigeonhole principle, this implies the existence of two distinct input sequences that the model generates the same output sequence. According to regularity assumption, there is an input sequence that the model generates an incorrect output sequence. Therefore, we have determined that $D = \Omega(\frac{\sqrt{L}}{\ln L}) = \tilde{\Omega}(\sqrt{L})$. □

C. Proofs of the theorems in Section 5

In this section, we will prove the theorems and propositions outlined in Section 5. Firstly, we will provide a precise and formal definition of the problems and statements for each theorem and proposition. Subsequently, we will offer a thorough proof for each theorem and proposition.

This paper utilizes the identical formulation of the arithmetic evaluation task and CoT solution presented by Feng et al. (2023). The arithmetic evaluation task is denoted as $\mathsf{Arithmetic}(n, p)$ and is defined on the finite field modulo p, with the input length not exceeding n. The CoT solution can be formally defined as follows: for any arithmetic expression, there must exist a handle, which refers to a pair of adjacent numbers connected by an operator that can be evaluated. In the CoT method, we solve the leftmost variable first in each step and connect the equations of each step using the equal sign.

C.1. Proofs of Proposition 5.2

In this section, we will give a proof of Proposition 5.2.

Theorem C.1. *For any integer n, a log-precision linear Transformer with a constant depth M and a constant number of heads H can generate the correct output for the arithmetic evaluation task for all expressions of length no more than n only if the hidden dimension $D = \tilde{\Omega}(\sqrt[4]{L})$.*

Proof. Notice that the generated CoT sequence can be determined by the result of the first step. The different possibilities of results for the first step are at least $\Omega(\exp(n))$ since each a_1 OP_1 $a_2 \cdots$ are legal where $a_i \in \mathbb{F}_p$ and $\text{OP}_i \in \{+, -, \times, \div\}$. On the other hand, the hidden state corresponding to the input sequence determines the output CoT sequence. Thus, the size of hidden state should be at least $\Omega(n) = \Omega(\sqrt{L})$. This means that the hidden dimension should be at least $\Omega(\sqrt{\sqrt{L}/\log L}) = \tilde{\Omega}(\sqrt[4]{L})$, which ends our proof. □

C.2. Proofs of Proposition 5.4

In this subsection, we will prove Proposition 5.4, which is a corollary of the theorem from Feng et al. (2023) such that the sparse transformer can generate the correct output for the DP problem.

Theorem C.2 (From Feng et al. (2023))**.** *For any DP problem, any integer $n \in \mathbb{N}$, there exists an autoregressive Transformer with constant depth L, hidden dimension d and attention heads H (independent of n), such that the answer generated by the Transformer is correct for all input sequences $\boldsymbol{s}$ of length no more than n. Moreover, all parameter values are bounded by $O(\text{poly}(n))$.*

Theorem C.3. *Consider any m-locality DP problem satisfying the same condition as in Theorem 4.1. Given any integer $n > 0$, let L be the length of the output sequence when the input sequence length is n. Then, there exists a (log-precision) sparse Transformer with block size $B = \Theta(m)$, a constant depth M, a constant number of attention heads H, and a constant hidden dimension D that can generate the correct output for all inputs $\boldsymbol{s}$ of length n.*

Proof. Under the assumption of locality, we can treat the sparse transformer as a standard transformer to solve the m-locality DP problem. When the standard transformer solves the m-locality DP problem, the attention head will only attend to the tokens with the distance at most m, and the attention to other tokens is 0. The sparse transformer adds masks to other tokens, and therefore, is equivalent to the standard transformer for m-locality DP problem. According to Theorem C.3, the sparse transformer can solve m-locality DP problem. □

C.3. Proofs of Proposition 5.5

In this section, we will give a proof of Proposition 5.5, which is a corollary of Theorem 4.4.

Theorem C.4. *Consider any m-locality regular DP problem satisfying the same condition as in Theorem 4.1 and assume that $m = \Theta(n)$ where n is the input sequence length. Then, a log-precision linear Transformer with a constant depth M and a constant number of heads H can generate the correct output for all inputs $\boldsymbol{s}$ of length n only if the hidden dimension $D = \tilde{\Omega}(\sqrt{m})$.*

Proof. Same as the arguments used in the proof of Theorem 4.2, we can get the size of hidden states should be at least $\Omega(n) = \Omega(m)$. The regularity assumption implies that the hidden dimension should be at least $\Omega(\sqrt{m/\log m}) = \tilde{\Omega}(\sqrt{m})$, concluding our proof. □

D. Additional Experiments

The accuracies of standard and efficient Transformers with 5 layers on the ED task are shown in Figure 3. The results are similar to the results we obtained when using 3 layers. This evidence further supports our theoretical analysis.

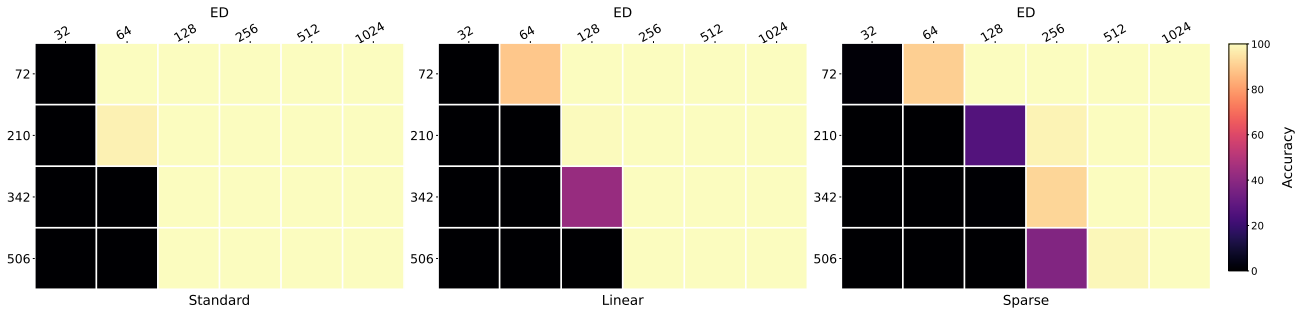

Figure 3. A comparison of accuracies on different model types with ED task. Each subplot corresponds to a model (Standard Transformer, Linear Transformer, Sparse Transformer). Within each subplot, the x-axis represents the embedding dimension, and the y-axis denotes the problem size. The color intensities indicate the accuracy level achieved by the respective models.