Transformers, Parallel Computation, and Logarithmic Depth

Clayton Sanford [1] **Daniel Hsu** [1] **Matus Telgarsky** [2]

Abstract

We show that a constant number of self-attention layers can efficiently simulate—and be simulated by—a constant number of communication rounds of *Massively Parallel Computation*, a popular model of distributed computing with wide-ranging algorithmic results. As a consequence, we show that logarithmic depth is sufficient for transformers to solve basic computational tasks that cannot be efficiently solved by several other neural sequence models and sub-quadratic transformer approximations. We thus establish parallelism as a key distinguishing property of transformers.

1 Introduction

The transformer (Vaswani et al., 2017) has emerged as the dominant neural architecture for many sequential modeling tasks such as machine translation (Radford et al., 2019) and protein folding (Jumper et al., 2021). Reasons for the success of transformers include suitability to modern hardware and training stability: unlike in recurrent models, inference and training can be efficiently parallelized, and training is less vulnerable to vanishing and exploding gradients. However, the advantages of transformers over other neural architectures can be understood more fundamentally via the lens of *representation*, which regards neural nets as parameterized functions and asks what they can efficiently compute.

Many previous theoretical studies of transformers establish (approximation-theoretic and computational) universality properties, but only at large model sizes (Yun et al., 2020; Pérez et al., 2021). These results are not unique to transformers and reveal little about which tasks can be solved in a *size-efficient* manner. Several other works (e.g., Hahn, 2020; Merrill & Sabharwal, 2022; Sanford et al., 2023) give fine-grained representational results in the scaling regime where context length grows but model depth is constant. In this regime, basic algorithmic tasks like matching parentheses and evaluating Boolean formulas are impossible.

In this work, we identify parallelism as a key to distinguishing transformers from other architectures. While recurrent architectures process their inputs serially, transformers allow independent interactions between the input tokens, mediated by the inner products between query and key embeddings in self-attention units. We leverage this property of self-attention to establish a formal connection between transformers and *Massively Parallel Computation (MPC)* (Karloff et al., 2010). Concretely, we design transformers that simulate MPC protocols (and vice versa), and in doing so, we exhibit a wide range of computational tasks that are solved by logarithmic-depth transformers, including tasks that cannot be efficiently solved with other architectures such as graph neural nets and recurrent models.

1.1 Our Results

We advance the understanding of transformers' representational capabilities with the following results.

1. The algorithmic capabilities and limitations of logarithmic-depth transformers are captured by the MPC model (Section 3).
2. There is a simple sequential task that (i) is solved by (and, empirically, learned from data using) logarithmic-depth transformers, but (ii) *cannot* be efficiently solved by several alternative architectures (Sections 4 and 5).

In more detail, our first collection of results, Theorems 3.1 and 3.4, show that any R-round MPC protocol can be implemented by a transformer of depth $O(R)$, and that any depth-L transformer can be simulated by an $O(L)$-round MPC protocol. The former implies that several graph problems are solved by logarithmic-depth transformers (Corollary 3.3); the latter implies the near-optimality of these transformers (Corollary 3.5) conditional on a well-known conjecture about the limitations of MPC algorithms (Conjecture 2.4). A key technical step (Lemma 3.2) shows how transformers can implement the simultaneous message-passing used in MPC protocols to communicate between machines. While previous works (Sanford et al., 2023) have used communication

[1]Department of Computer Science, Columbia University, New York, NY, USA [2]Courant Institute, New York University, New York, NY, USA. Correspondence to: Clayton Sanford <clayton.h.sanford@gmail.com>.

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

complexity to understand the representational limitations of self-attention layers, our results show the benefits of the communication lens for understanding the strengths of transformers as well.

Our second set of results concern the k*-hop induction heads* task, a synthetic sequential task that draws inspiration from the induction heads primitive of Elhage et al. (2021). The theoretical results of Section 4 prove that depth $L = \Theta(\log k)$ is necessary and sufficient for efficient transformer representation. An accompanying empirical investigation reveals that transformers trained on the task obey the same threshold and recover a similar model to the theoretical construction. In contrast, Section 5 illustrates that non-parallelizable recurrent architectures—including state-space models like Mamba (Gu & Dao, 2023)—are unable to solve the task in a size-efficient manner. Moreover, well-known transformer models with computationally-efficient alternatives to self-attention, like Performer (Choromanski et al., 2022) and Longformer (Beltagy et al., 2020), and shallow transformers with chain-of-thought prompting sacrifice their abilities to implement parallel algorithms, as evidenced by their proven inability to solve this task.

1.2 Related Work

Some of the types of lower bounds we sought in this work were inspired by the literature on depth-separation for feed-forward neural networks (e.g., Eldan & Shamir, 2016; Daniely, 2017; Telgarsky, 2016), which exhibit functions that are efficiently approximated by deep networks, but not by shallower networks.

Many theoretical approaches have been used to understand the representational capabilities of transformers and self-attention units in various scaling regimes. Some works model (variants of) transformers as machines for recognizing formal languages, such as the Dyck languages (Hahn, 2020; Bhattamishra et al., 2020; Yao et al., 2021; Hao et al., 2022) and star-free regular languages (Angluin et al., 2023). These approaches reveal inability of fixed-size transformers to handle arbitrarily long inputs. Other works show how transformers can simulate finite-state automata (Liu et al., 2022) with logarithmic depth, and Turing machines with (unrolled) depth (or chain-of-thought length) scaling polynomially with total runtime (Wei et al., 2021; Malach, 2023; Merrill & Sabharwal, 2023b). However, it is unclear if these results are near optimal or even transformer-specific.

Theoretical results about the limitations of constant-depth transformers have been articulated by way of analogy to circuit complexity (Merrill & Sabharwal, 2023a; Merrill et al., 2022; Merrill & Sabharwal, 2022; Strobl, 2023; Strobl et al., 2023), implying the inability of constant-depth transformers to solve tasks like graph connectivity and Boolean formula evaluation. Other works characterize the representational capabilities of one-layer transformers (Likhosherstov et al., 2021; Sanford et al., 2023), but these approaches do not apply to deeper models. Sanford et al. study multi-headed attention using communication complexity, a framing that informs this work's connection to distributed computing.

The MPC model (Karloff et al., 2010; Beame et al., 2017; Goodrich et al., 2011; Andoni et al., 2014; Im et al., 2023) was introduced to study distributed computing frameworks such as MapReduce (Dean & Ghemawat, 2004). A major goal is to design protocols that use few rounds of communication for setups in which each machine's local memory is sublinear in the input size. Many advances have been made in MPC algorithms for important problems, including weighted interval selection, approximate maximum matching, and clustering (see, e.g., Im et al., 2023, for a recent survey). Nanongkai & Scquizzato (2022) established equivalence classes among MPC algorithmic tasks, proving that determining the connectivity of a graph is equivalent to numerous other graph reasoning tasks—such as bipartiteness testing and cycle detection—in $O(1)$ rounds of MPC computation.

The centrality of graph connectivity to the study of MPC is further evident in its conjectured hardness. Connectivity in sparse graphs is a basic problem that has resisted progress, and all all MPC protocols in this memory regime appear to require $\Omega(\log n)$ rounds for input graphs on n vertices. Lower bounds in MPC and related models were studied by Beame et al. (2017), Roughgarden et al. (2018), and Charikar et al. (2020). The conjectured impossibility of $o(\log n)$-round protocols for connectivity is now used as basis for conditional lower bounds (Ghaffari et al., 2019).

Simulation of transformers by recurrent models (Oren et al., 2024) and simulation of graph neural nets (GNNs) by transformers (Kim et al., 2022) offer some coarse-grain insight into the relationship between these architectures, but separations are not implied by these previous works. Our connection between transformers and MPC is most similar to that established by Loukas (2019) between GNNs and the CONGEST model of distributed computation. Both works establish positive and negative results by identifying neural architectures with communication protocols. In Section 5.1, we show that the MPC connection allows transformers solve graph connectivity more efficiently than GNNs.

Our k-hop induction heads task is designed as a k-fold composition of its standard analogue (Elhage et al., 2021). It is similar to a special case of the LEGO reasoning task (Zhang et al., 2023), which reveals the super-linear benefit of depth with respect to k; in our case, we theoretically and empirically exhibit an exponential benefit. We also draw a connection to the well-studied problem of pointer-chasing (Papadimitriou & Sipser, 1982; Duris et al., 1984; Nisan & Wigderson, 1993), which enables the proof of our separation between parallel and serial architectures. Our fine-grained empirical interpretability analysis for synthetic tasks draws inspiration from similar approaches for the analysis of sequential algorithms like sorting and reversal (Li & McClelland, 2022).

2 Preliminaries

2.1 Transformers

We first define a self-attention head, the core primitive of a transformer. The *softmax* operator is $\mathrm{softmax}(v) = (\exp(v_1), \dots, \exp(v_N))/\sum_{j=1}^N \exp(v_j)$ for $v \in \mathbb{R}^N$. We apply softmax to matrices $A \in \mathbb{R}^{N\times N}$ row-wise, i.e. $\mathrm{softmax}(A)_i = \mathrm{softmax}((A_{i,1}, \dots, A_{i,N}))$.

Definition 2.1 (Self-attention head)**.** A *self-attention head* is a mapping $f_{Q,K,V} : \mathbb{R}^{N\times m} \to \mathbb{R}^{N\times m}$ defined by

$$f_{Q,K,V}(X) = \mathrm{softmax}(Q(X)K(X)^\mathsf{T})V(X)$$

and parameterized by row-wise *query*, *key*, and *value embeddings* $Q, K, V \colon \mathbb{R}^{N\times m} \to \mathbb{R}^{N\times m}$ (e.g., $Q(X) = (Q_1(X_1), \dots, Q_N(X_N))$). Let Attn_m^N denote the set of all self-attention heads with embedding dimension m and context length N.

A transformer composes L layers of H self-attention heads per layer, plus an output multi-layer perceptron (MLP).

Definition 2.2 (Transformer)**.** A *transformer* is a mapping $T : \mathbb{R}^{N\times d_{\text{in}}} \to \mathbb{R}^{N\times d_{\text{out}}}$ specified by self-attention heads $(f_{\ell,h} \in \mathsf{Attn}_m^N)_{\ell\in[L],h\in[H]}$ and element-wise input and output MLPs:

$$\phi = (\phi_1, \dots, \phi_N),\ \psi = (\psi_1, \dots, \psi_N) : \mathbb{R}^{N\times m} \to \mathbb{R}^{N\times d_{\text{out}}}.$$

Upon input $X \in \mathbb{R}^{N\times d_{\text{in}}}$, the transformer computes intermediate embeddings $X^0, \dots, X^L \in \mathbb{R}^{N\times m}$ with $X^0 = \phi(X)$ and

$$X^\ell = X^{\ell-1} + \sum\nolimits_{h=1}^H f_{\ell,h}(X^{\ell-1}),$$

and returns $T(X) = \psi(X^L)$ as output. Let $\mathsf{Transformer}_{m,L,H,d_{\text{in}},d_{\text{out}}}^N$ denote the set of all such transformers, and $\mathsf{Transformer}_{m,L,H}^N := \mathsf{Transformer}_{m,L,H,1,1}^N$.

Modeling assumptions. We treat the transformer as a computational model that permits arbitrary element-wise computation, but restricts the manner in which multiple elements are processed together. This manifests in our decisions to model query/key/value embeddings and MLPs as arbitrary functions on the embedding space; Loukas (2019) employs a similar modeling assumption for GNNs. Note that the element-wise embeddings and MLPs may be index-specific, obviating the need for positional embeddings.

Our theoretical results cover the scaling regime where the context length N is the main asymptotic parameter; while the embedding dimension m, the number of heads H, and the depth L grow sub-linearly in N. This reflects real-world trends in large-language models, where context length has sharply increased in recent years.

Throughout, we assume all intermediate computations in transformers are represented by p-bit precision numbers for $p = \Theta(\log N)$. Limiting the precision is consistent with recent practice of using low-precision arithmetic with transformers (e.g., Wang et al., 2022; Dettmers et al., 2022). We discuss this precision assumption in greater detail in Appendix A.1, along with other minor technical assumptions (such as the inclusion of a “start token” for mathematical convenience).

Masked Transformers. We also consider *masked self-attention*, where only certain inner products influence the softmax output. Let $\Lambda \in \{-\infty, 0\}^{N\times N}$ be a *masking matrix* with at least one zero entry in every row. Then, a Λ-*masked self-attention* unit is defined by

$$f_{Q,K,V}^\Lambda(X) = \mathrm{softmax}(Q(X)K(X)^\mathsf{T} + \Lambda)V(X).$$

Let $\Lambda\text{-}\mathsf{Attn}_m^N$ and $\Lambda\text{-}\mathsf{Transformer}_{m,L,H}^N$, respectively, denote the sets of all Λ-masked self-attention heads and all transformers comprised of those heads. We define *causally-masked transformers* by $\mathsf{MaskAttn}_m^N := \Gamma\text{-}\mathsf{Attn}_m^N$ and $\mathsf{MaskTransformer}_{m,L,H}^N := \Gamma\text{-}\mathsf{Transformer}_{m,L,H}^N$, where Γ is the lower-triangular mask with $\Gamma_{i,j} = 0$ iff $i \geq j$.

2.2 Massively Parallel Computation Model

We use the definition of MPC from Andoni et al. (2018).

Definition 2.3 (MPC protocol)**.** For any global and local memory constants $\gamma, \delta > 0$, a (γ, δ)-*MPC protocol* for a function $f : \mathbb{Z}_{2^p}^{n_{\text{in}}} \to \mathbb{Z}_{2^p}^{n_{\text{out}}}$ specifies a distributed computing protocol for $q = \Theta(n_{\text{in}}^{1+\gamma-\delta})$ machines, each with $s = O(n_{\text{in}}^\delta)$ words[1] of local memory to jointly compute $f(\mathtt{Input})$ for any given $\mathtt{Input} \in \mathbb{Z}_{2^p}^{n_{\text{in}}}$ as follows. The $\mathtt{Input} \in \mathbb{Z}_{2^p}^{n_{\text{in}}}$ is distributed across the local memories of the first $\lceil n_{\text{in}}/s \rceil$ machines. Computation proceeds in rounds. In each round, each machine computes an arbitrary function of its local memory to prepare at most s words to send to other machines; messages are simultaneously transmitted, and the protocol ensures that each machine receives at most s words at the end of the round. After the final round, the $\mathtt{Output} = f(\mathtt{Input}) \in \mathbb{Z}_{2^p}^{n_{\text{out}}}$ is in the local memories of the first $\lceil n_{\text{out}}/s \rceil$ machines. See Figure 1 for details.

Our negative results in Section 3.2 are conditional on the well-known “one-versus-two cycle” conjecture (Beame

[1] We assume the word size is $p = \Theta(\log n_{\text{in}})$ bits. For convenience, we regard words as elements of $\mathbb{Z}_{2^p}$ (integers mod 2^p).

- $\texttt{Input}{=}(\texttt{Input}_1, \dots, \texttt{Input}_{n_{\text{in}}}) \in \mathbb{Z}_{2^p}^{n_{\text{in}}}$ is distributed across local memories of machines $1 \leq i \leq \lceil \frac{n_{\text{in}}}{s} \rceil$:

$$\texttt{MachineIn}_i^{(1)} = \{(\texttt{Input}_\iota, \iota) : \iota \in \{(s{-}1)i{+}1, \dots, \min\{n_{\text{in}}, si\}\}\}.$$

- For round $r = 1, \dots, R$:
  - Each machine $i \in [q]$ computes messages $(\texttt{MsgOut}_{i,j}^{(r)})_{j=1,2,\dots}$ to send to machines $(\texttt{Dest}_{i,j}^{(r)})_{j=1,2,\dots}$ as function of $\texttt{MachineIn}_i^{(r)}$: $\texttt{MachineOut}_i^{(r)} = \texttt{Local}_{r,i}(\texttt{MachineIn}_i^{(r)}) = \{(\texttt{MsgOut}_{i,j}^{(r)}, \texttt{Dest}_{i,j}^{(r)}) \in \mathbb{Z}_{2^p}^{d_j} \times [q] : j = 1, 2, \dots\}$; $\sum_j d_j \leq s$ is ensured.
  - All messages are simultaneously transmitted; the messages in local memory of machine $i \in [q]$ for round $r + 1$ are:

$$\texttt{MachineIn}_i^{(r+1)} = \{(\texttt{Msg}, \texttt{Src}) : (\texttt{Msg}, i) \in \texttt{MachineOut}_{\texttt{Src}}^{(r)}\}; \quad \sum_{(\texttt{Msg},\texttt{Src}) \in \texttt{MachineIn}_i^{(r+1)}} |\texttt{Msg}| \leq s \text{ is ensured.}$$

- $\texttt{Output}{=}f(\texttt{Input})$ comes from $\texttt{MachineIn}_i^{(R+1)} = \{(\texttt{Output}_\iota, \texttt{Src}) : \iota \in \{(s{-}1)i{+}1, \dots, \min\{n_{\text{out}}, si\}\}\}$ for $1 \leq i \leq \lceil \frac{n_{\text{out}}}{s} \rceil$.

Figure 1. Formal execution of an MPC protocol for computing $f\colon \mathbb{Z}_{2^p}^{n_{\text{in}}} \to \mathbb{Z}_{2^p}^{n_{\text{out}}}$. ($|\texttt{Msg}|$ is the number of words in $\texttt{Msg}$.)

et al., 2017; Roughgarden et al., 2018; Ghaffari et al., 2019).

Conjecture 2.4 (see, e.g., Ghaffari et al., 2019)**.** *For any $\gamma > 0$, $\delta < 1$, and N, if π is an (γ, δ)-MPC protocol that distinguishes a single cycle on N nodes and a union of two cycles each on $N/2$ nodes, then π uses $\Omega(\log N)$ rounds.*

2.3 Graphs as Sequential Inputs

When providing a graph $G = (V, E)$ as input to transformers or MPC protocols, we serialize G as a sequence in $[|V|]^{2|E|}$ that encodes each edge as a pair of vertex tokens. The resulting transformer has $N = 2|E|$ and $d_{\text{in}} = 1$, and the resulting MPC protocol has $n_{\text{in}} = 2|E|$.

3 Relating Transformers and MPC

We coarsely characterize the computational power of transformers in a certain size regime by establishing a bidirectional relationship between transformers and MPC. Theorems 3.1 and 3.4 show that any MPC protocol can be simulated by a transformer, and vice versa. As corollaries (Corollaries 3.3 and 3.5), we obtain tight upper and lower bounds on the depth of bounded-size transformers for computing connected components in graphs.

3.1 Simulation of MPC Protocols by Transformers

The following theorem shows that any MPC protocol π with sublinear local memory can be simulated by a transformer whose depth L is linear in the number of rounds R of π, and embedding dimension m is polynomial in the local memory size $s = O(N^\delta)$ of machines used by π.

Theorem 3.1. *For constants $0 < \gamma < \delta < 1$ and any deterministic R-round (γ, δ)-MPC protocol π on n_{in} input words and $n_{\text{out}} \leq n_{\text{in}}$ output words, there exists a transformer $T \in \mathsf{Transformer}_{m,L,H}^N$ with $N = n_{\text{in}}, m = O(n_{\text{in}}^{4\delta} \log n_{\text{in}}), L = R + 1, H = O(\log \log n_{\text{in}})$ such that $T(\texttt{Input})_{:n_{\text{out}}} = \pi(\texttt{Input})$ for all $\texttt{Input} \in \mathbb{Z}_{2^p}^N$.*

The theorem provides a non-trivial construction in the strongly sub-linear local memory regime when $s = O(N^{1/4-\epsilon})$ for any $\epsilon > 0$.[2] Because numerous tasks, including approximate maximum matching, submodular maximization, and weighted interval selection, can be solved by MPC protocols with $O(N^\delta)$ memory for any fixed $\delta \in (0, 1)$ (Ghaffari, 2019), these tasks are similarly implementable by transformers with sub-linear embedding dimension. Subsequent work by Sanford et al. (2024) improves this analysis by proving that any MPC protocol with local memory $s = O(N^{1-\epsilon})$ for any $\epsilon \in (0, 1)$ can be simulated by a transformer of embedding dimension $m = O(N^{1-\epsilon'})$ for some $\epsilon' \in (\epsilon, 1)$.

Theorem 3.1 Proof Overview. At a high level, the proof in Appendix B.2 entails simulating each round of parallel computation with a single-layer transformer and applying those constructions serially to $\texttt{Input}$. The local computation on each machine (represented by $\texttt{MachineOut}_i^{(r)} = \texttt{Local}_{r,i}(\texttt{MachineIn}_i^{(r)})$) is directly encoded using element-wise query/key/value embeddings.

The crux of the proof involves the simulation of a *routing protocol* to determine $\texttt{MachineIn}^{(r+1)}$ from $\texttt{MachineOut}^{(r)}$. We construct a self-attention unit that ensures that an encoding of a sequence of addressed messages from each machine are properly routed to their destinations.[3]

For any message size β, message count bound s, and number of tokens N, we say that $(\texttt{Sent}, \texttt{Rcvd}) \in \mathbb{R}^{N \times m} \times \mathbb{R}^{N \times m}$ is a *valid* (β, s)*-routing* if, for each $i \in [N]$, the i-th row of $\texttt{Sent}$ (resp. $\texttt{Rcvd}$) is the vector encoding of some $\texttt{Sent}_i \subset \mathbb{Z}_{2^p}^\beta \times [N]$ (resp. $\texttt{Rcvd}_i \subset \mathbb{Z}_{2^p}^\beta \times [N]$) such that

$$\texttt{Rcvd}_i = \{(\texttt{Msg}, \texttt{Src}) : (\texttt{Msg}, i) \in \texttt{Sent}_{\texttt{Src}}\},$$

[2] Applying Theorem 3.1 when $\delta \geq \frac{1}{4}$ yields transformers with embedding dimension $m \geq N$, which trivializes the transformer architecture and negates any advantages of depth under our MLP universality assumption. This is due to the fact a transformer with N-dimensional embeddings could aggregate the entire input sequence $X \in \mathbb{R}^N$ in a single embedding and use its output MLP to compute any arbitrary function on that input.

[3] This routing between machines uses the all-pairs structure of self-attention and may not admit a subquadratic approximation.

and each of $\texttt{Rcvd}_i$ and $\texttt{Sent}_i$ has cardinality at most s.[4]

Lemma 3.2. *For any $\beta, s, N \in \mathbb{N}$, there exists a transformer $\mathrm{route}_{\beta,s} \in \mathsf{Transformer}_{m,1,1}^{N}$ with $m = O(s^4 \beta \log N)$ satisfying $\mathrm{route}_{\beta,s}(\texttt{Sent}) = \texttt{Rcvd}$ for any valid (β, s)-routing $(\texttt{Sent}, \texttt{Rcvd})$.*

The proof of Lemma 3.2 appears in Appendix B.1 and combines two key techniques: sparse propagation and multiple hashing. The former is a simple variant of the "sparse averaging" task of Sanford et al. (2023), which simultaneously computes N averages over subsets of inputs; this task is solved a single self-attention head with small embedding dimension (Proposition B.1). Using sparse propagation, we construct a self-attention head that averages the $\leq s$ encodings of each $\texttt{Rcvd}_{\texttt{Src}}$ for every $\texttt{Src} \in \texttt{Rcvd}_i$. In order to ensure that we can decode that average of encodings, we apply error-correction by encoding each $\texttt{Output}_i$ in a sparse and redundant manner, where each outgoing messages appears as multiple copies of the same addressed "packet."

Application: Connectivity with Log-Depth Transformers. As an immediate consequence of Theorem 3.1, any graph problem solvable with a logarithmic number of rounds of MPC computation (and local memory s) is also computable by a logarithmic depth transformer (and embedding dimension $\tilde{O}(s^4)$). The following result—which bounds transformer depth needed to compute connected components of a graph G—follows from Theorem 6.2 of Coy & Czumaj (2022), which derandomizes an MPC algorithm of Behnezhad et al. (2019), and Theorem 3.1.

Corollary 3.3. *For any constant $\epsilon \in (0, 1)$ and any $D \leq N$, there exists a transformer in $\mathsf{Transformer}_{m,L,H}^{N}$ with $m = O(N^\epsilon)$, $H = O(\log \log N)$, and $L = O(\log D)$ that identifies the connected components of any input graph $G = (V, E)$ with $|V|, |E| = O(N)$ where each connected component has diameter at most D.*

Coy & Czumaj also give efficient MPC algorithms for other related problems (e.g., spanning forest), so we obtain efficient transformers for these problems, too (Appendix B.3).

3.2 Simulation of Transformers by MPC protocols

The following theorem shows that MPC protocols can simulate transformers and prove depth lower bounds on transformers, conditioned on Conjecture 2.4. We get, as a corollary, the conditional optimality of the transformer depth bound in Corollary 3.3.

Theorem 3.4. *For any transformer $T \in \mathsf{Transformer}_{m,L,H}^{N}$ (or Λ-$\mathsf{Transformer}_{m,L,H}^{N}$) with $mH = O(N^\delta)$ for $\delta \in (0, 1)$ and any $\delta' \in (\delta, 1)$, there exists a $O(\frac{L}{\delta' - \delta})$-round $(1 + \delta', \delta')$-MPC protocol with $q = O(N^2)$ machines with $s = O(N^{\delta'})$ local memory for computing T.*

Theorem 3.4 demonstrates that the algorithmic capabilities of transformers are no stronger than those of MPC protocols with a quadratic scaling in the number of machines. While Theorems 3.1 and 3.4 do not jointly provide a sharp characterization of the two computational models, the reductions are tight enough to provide strong evidence for the optimality of the connected components construction of Corollary 3.3.

Theorem 3.4 Proof Overview. At a high-level, the proof in Appendix C.1 constructs an MPC protocol that simulates a self-attention layer by separating the computation of MLPs and attention matrices into three separate categories of machines.

- Each input token is provided to its own *token machine*, responsible for preparing the query/key/value embeddings.
- Each pair of tokens is associated with an *inner product machine* that will compute the inner product between their respective query and key embeddings.
- *Propagation machines* ensure that embeddings are routed to the proper inner product machine and compute outputs of each softmax unit.

The proof gives the communication protocol for these machines, shows how they simulate a layer of self-attention in $O(1/(\delta' - \delta))$ rounds, and establishes the sufficiency of $O(N^2)$ machines with $O(N^{\delta'})$ local memory.

Application: Conditional Optimality of Corollary 3.3. Assuming the well-established Conjecture 2.4, we prove an $\Omega(\log D)$ lower bound on the depth of parameter-efficient transformers for determining connectivity of graphs where connected components may have diameter up to D.

Corollary 3.5. *Let $\epsilon \in (0, 1)$ be any constant, and let $D \geq N^\epsilon$. Assume Conjecture 2.4, and suppose there exists $T \in \mathsf{Transformer}_{m,L,H}^{N}$ with $mH = O(D^{1-\epsilon})$ that decides connectivity of any input graph with connected components having diameter $\leq D$. Then $L = \Omega(\log D)$.*

4 Transformers for k-Hop Induction Heads

We complement the generality of Section 3 by studying, both empirically and theoretically, a specific toy sequential modeling task which will also serve (in Section 5) as a problem to separate the representational capabilities of transformers from that of other neural architectures.

This task, called the *k-hop induction heads* task, draws inspiration from the original *induction heads* task defined and analyzed on trained language models and in synthetic environments by Elhage et al. (2021) (see also Bietti et al.,

[4] We abuse notation by writing $\texttt{Dest} \in \texttt{Sent}_i$ to mean there exists some $\texttt{Msg}$ such that $(\texttt{Msg}, \texttt{Dest}) \in \texttt{Sent}_i$.

2023). The standard induction heads task completes bigrams auto-regressively by predicting the token that follows the last previous occurrence of the final token in the sequence. For example, given the input $X = \texttt{baebcabebdea}$, the standard induction heads task is to complete the final bigram by predicting $\texttt{b}$ for the final token.

The k-hop induction heads tasks generalizes this mechanism by repeatedly using the completion of a bigram to determine the next bigram to complete. In the previous example, the 2-hop induction heads task is to predict $\texttt{c}$ for the final token:

$$\texttt{baebcabebdea}.$$

Definition 4.1. For any finite alphabet Σ, define the map $\mathrm{hop}_k \colon \Sigma^N \to (\Sigma \cup \{\perp\})^N$ by $\mathrm{hop}_k(X)_i = X_{\mathrm{find}_X^k(i)}$ if $\mathrm{find}_X^k(i) \neq 0$ and $\perp$ otherwise, where

$$\mathrm{find}_X^1(i) = \max(\{0\} \cup \{j \in \mathbb{N} : j \leq i,\, X_{j-1} = X_i\});$$
$$\mathrm{find}_X^k(i) = \mathrm{find}_X^1(\mathrm{find}_X^{k-1}(i)) \quad \text{for } k \geq 2.$$

The *k-hop induction heads task* is to compute, for each $i = 1, \dots, N$, the value of $\mathrm{hop}_k(X)_i$ from $(X_1, \dots, X_i)$.

We note a similarity to the LEGO tasks of (Zhang et al., 2023), who empirically study the ability of transformers to learn sequential operations on Abelian groups and observe the ability to perform more operations than the depth of the network.

4.1 Log-Depth Transformer for k-Hop Induction Heads

Although hop_k appears to requires k steps to solve, we show that it is solved by a transformer of depth $O(\log k)$.

Theorem 4.2. *For any $k \in \mathbb{N}$ and alphabet Σ with $|\Sigma| \leq N$, there exists $T \in \mathsf{MaskTransformer}_{m,L,H}^N$ that computes $\mathrm{hop}_k \colon \Sigma^N \to (\Sigma \cup \{\perp\})^N$ with $m = O(1)$, $L = \lfloor \log_2 k \rfloor + 2$, and $H = 1$.*

In contrast to Corollary 3.3, this construction has constant embedding dimension and is achieved by a causally-masked transformer. As such, its proof in Appendix D.1 depends on other techniques that exploit the simplicity of the problem and build on the induction heads construction of Bietti et al. (2023), rather than simply applying Theorem 3.1.

We give evidence for the optimality of this construction by proving a conditional lower bound using Theorem 3.4, as was done in Corollary 3.5.

Corollary 4.3. *Assuming Conjecture 2.4, for any constants $\xi \in (0, 1/2]$ and $\epsilon \in (0, 1)$, and any even $k = \Theta(N^\xi)$, every transformer $T \in \mathsf{MaskTransformer}_{m,L,H}^N$ with $mH = O(k^{1-\epsilon})$ that computes hop_k has depth $L = \Omega(\log k)$.*

4.2 Log-Depth Transformer Learned from Data

We empirically assess whether the representational tradeoffs elucidated by tasks efficiently solved by parallelizable algorithms have implications for optimization and generalization properties of transformers. To that end, we trained auto-regressive transformer architectures of varying sizes to solve $\mathrm{hop}_k(X)$ for a variety of values of k in order to understand how changing depth impacted the performance of the learned models, the goal being to verify the sufficiency of logarithmic depth, just as in our theory.

In brief, we trained transformers with 500K to 5M parameters and depths $\{2, 3, 4, 5, 6\}$ with Adam to solve $\mathrm{hop}_k(X)$ for $k \in \{0, \dots, 16\}$ with context length $|N| = 100$ and alphabet size $|\Sigma| = 4$. We trained the transformers in a multi-task setting, where a single model was trained to predict the sequence $\mathrm{hop}_k(X)$ auto-regressively when provided with X and k drawn at random. Further experimental details can be found in Appendix G.1, and the experimental code is available at `https://github.com/chsanford/hop-induction-heads`.

We found that transformers are indeed capable of learning hop_k given sufficient training time, and that the largest learnable k grows exponentially with the depth. As can be seen in Figure 2, a six-layer neural network performs well on all $k \leq 16$, a five-layer on $k \leq 8$, a four-layer on $k \leq 4$, and so forth. We further explore these experimental results in Appendix G.2 and observe a performance threshold appears to specifically lie at $\lfloor \log_2 k \rfloor + 2$ that coincides with Theorem 4.2. This logarithmic dependence of the depth on k persists in a larger-width regime, which is explored in Appendix G.3. In the finite sample regime where neural networks are prone to overfit, our investigations in Appendix G.5 note improved generalization in deeper models, which suggests that deeper models have a favorable inductive bias for tasks like hop_k.

Moreover, the learned models are surprisingly interpretable. We examined the activation patterns of attention matrices, and found close correspondences to useful intermediate products such as find_X^j. Taken together, these indicate that the learned models mechanistically resemble the construction employed in the proof of Theorem 4.2. See Appendix G.4 for our investigation of model interpretability.

5 Separations between Transformers and Alternative Architectures

Sections 3 and 4 characterize the representational capability of transformers by providing algorithmic problems they can solve with logarithmic depth and small polynomial or constant width. In contrast, other well-known architectures are unable to solve those same problems in a parameter-efficient

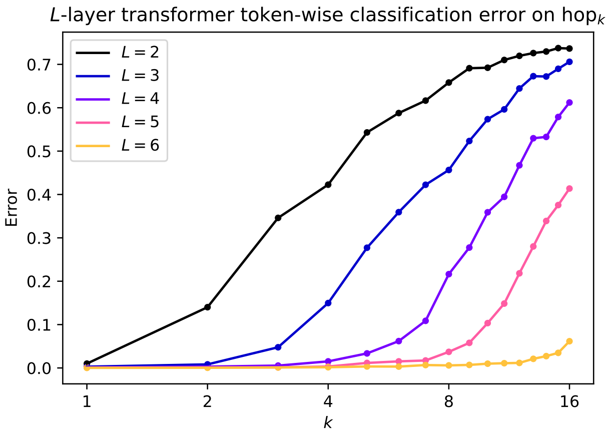


Figure 2. Evaluation of transformers of depths $L \in \{2,3,4,5,6\}$ trained on a mixture of hop_k for $k \in \{0,\dots,16\}$ evaluated on $n = 100$ samples of size $N = 100$ from each hop_k. Incrementing depth approximately doubles the largest k such that hop_k is learnable with small error.

manner. This section provides lower bounds on the parameter complexity of graph neural networks (GNNs), recurrent neural architectures, transformers with computationally efficient alternatives to softmax self-attention, and single-layer transformers with autoregressive chain-of-thought tokens needed to solve graph connectivity and the k-hop task.

5.1 GNNs Need Polynomial Depth for Graph Connectivity

The bidirectional relationship between transformers and MPC draws inspiration from past work drawing a similar connection between message passing graph neural networks (GNN_{mp}) and the CONGEST distributed computing model (Loukas, 2019). Their computation model of GNN_{mp} for width m and depth L closely resembles our $\mathsf{Transformer}_{m,L,H}^N$ in providing a general framework for the analysis of graph neural networks by allowing unbounded computation in each vertex with bounded communication on edges. On some input graph G, vertices send neighbors messages of size at most m—which are aggregated and crafted into new messages with MLPs—over L rounds of communication.

By restating Corollary 4.2 of (Loukas, 2019), we demonstrate a sharp contrast in the abilities of GNNs and transformers to solve graph algorithmic tasks.

Theorem 5.1 (Corollary 4.2 of (Loukas, 2019))**.** *There exists a graph G with N edges such that any GNN_{mp} with width m and depth L that determines whether an input subgraph H either (1) is connected or (2) forms a spanning tree of G requires $L\sqrt{m} = \tilde{\Omega}(N^{1/4})$.*

While Corollaries 3.3 and B.8 demonstrate the ability of transformers to determine whether any input graph is connected[5] or to identify a spanning tree with logarithmic depth and small polynomial width (i.e. $m = O(N^\epsilon)$), GNNs require depth $L = \tilde{\Omega}(N^{1/4-\epsilon/2})$ in the same regime. This gap is explainable by the fact that transformers on graph inputs G are not bound to pass messages exclusively along the edges of G. By “rewiring” the graphical structure in each layer, transformers can perform aggregation and “pointer passing” tasks with greater parametric ease than GNNs.

5.2 Suboptimality of Recurrent Architectures for hop_k

The logarithmic-depth and constant-width transformer implementation of hop_k in Theorem 4.2 cannot be replicated by recurrent neural architectures (Chung et al., 2014; Bengio et al., 1994; Turkoglu et al., 2021), including not just multi-layer recurrent neural networks (RNNs) but any sequential prediction procedure equivalent to them at inference time, which includes state space models such as Mamba (Gu & Dao, 2023).

We first consider a family of multi-layer RNNs of depth L and width m, consisting of arbitrary MLP units $g_\ell : \mathbb{R}^{m\times m} \to \mathbb{R}^{m\times m}$, which on input $X \in \mathbb{R}^{N\times d_{\text{in}}}$ produce output $Y \in \mathbb{R}^{N\times d_{\text{out}}}$ as follows using intermediates $X = Z^0, Z^1, \dots, Z^{L-1}, Z^L = Y \in \mathbb{R}^{N\times m}$[6]and hidden states $H^1, \dots, H^L \in \{0,1\}^{N\times m}$ with $H_0^\ell = \vec{0}$:

$$(Z_i^\ell, H_i^\ell) = g_\ell(Z_i^{\ell-1}, H_{i-1}^\ell),\ \forall i \in [N], \ell \in [L].$$

We provide a polynomial bound on the width and depth of a multi-layer RNN solving hop_k.

Corollary 5.2. *A multi-layer RNN of depth L and width m as above with $Y_N = \mathrm{hop}_k(X)_N$ satisfies either $L \geq k$ or $m = \Omega(\frac{N}{k^6})$.*

In contrast to Theorem 4.2, which demonstrates that depth $O(\log k)$ transformers with constant width suffice to solve hop_k for any k, Corollary 5.2 demonstrates that all multi-layer RNNs with width $O(N^{1/7})$ require depth k when $k = O(N^{1/7})$.

Mamba (Gu & Dao, 2023) can be seen as the combination of three ideas: (1) a continuous-time dynamics model of sequential prediction, powerful enough to model Kalman filters, hidden markov models, and many others; (2) a family of time-discretization schemes; (3) an unrolling technique to enable efficient linear-time training, using ideas similar to

[5] While the problem of subgraph connectivity for GNNs may at first glance appear more difficult than general graph connectivity for transformers, an implementation of this exact task can be implemented by modifying the protocol Corollary 3.3 to remove all edges from the graph that do not belong to H.

[6] We assume that $d_{\text{in}}, d_{\text{out}} \leq m$ and treat X and Y as if they are padded with zeros.

FlashAttention (Dao et al., 2022). Ultimately, at inference time, the time-discretization step results in an RNN (see Gu & Dao, 2023, Algorithm 2 and Theorem 1), and is therefore directly handled by Corollary 5.2.

This corollary is a near immediate application of a communication complexity fact about the hardness of solving multi-player *pointer-chasing* problems with limited communication among players (Guha & McGregor, 2009; Assadi & N, 2021). We provide the communication model and this result in Appendix E.1, and the reductions necessary to prove the above hardness results in Appendix E.2.

5.3 Suboptimality of Sub-Quadratic Attention Transformers for hop_k

Due to the quadratic computational cost of computing the attention matrix $\mathrm{softmax}(Q(X)K(X)^T) \in \mathbb{R}^{N\times N}$ and the continued desire for ever-larger context lengths, there is substantial interest in improving the computational complexity of the transformer architecture while preserving its expressive capabilities and inductive biases. As a result, a rich literature has emerged that proposes computationally-efficient alternatives to standard softmax attention. In this section, we demonstrate how several representative examples of sub-quadratic attention mechanisms lose the ability to perform efficient parallel computation under a logarithmic-depth scaling.

Kernel-Based Sub-Quadratic Attention. One approach to computationally-efficient approximation of transformers are *kernel-based sub-quadratic attention* mechanisms such as Performer (Choromanski et al., 2022), and PolySketchformer (Kacham et al., 2023). Both approximate the attention matrix $\mathrm{softmax}(Q(X)K(X)^\mathsf{T})$ with a low-rank matrix $Q'(X)K'(X)^\mathsf{T}$ where $Q', K' : \mathbb{R}^m \to \mathbb{R}^{m'}$ are applied element-wise. For sufficiently small $m' \ll N$, $Q'(X)K'(X)^\mathsf{T}V(X)$ can be computed efficiently by first computing $K'(X)^\mathsf{T}V(X) \in \mathbb{R}^{m'\times m}$, bounding the total runtime as $O(Nmm')$, rather than $O(N^2 m)$.

Let $\mathsf{KernelFormer}^N_{m,m',L,H}$ denote all H-headed L-layer transformer whose softmax attention modules are replaced by kernel-based sub-quadratic attention. We demonstrate the limitations of $\mathsf{KernelFormer}^N_{m,m',L,H}$ by showing that, unlike $\mathsf{Transformer}^N_{m,L,H}$, they have no depth-efficient implementation of hop_k.

Corollary 5.3. *Any* $T \in \mathsf{KernelFormer}^N_{m,m',L,H}$ *with* $T(X)_N = \mathrm{hop}_k(X)_N$ *satisfies either* $L \geq k$ *or* $mm'Hp = \Omega(\frac{N}{k^6})$.

Under a parameter-efficient regime where $mpHL = O(N^\epsilon)$, solving hop_k for $k = \Theta(N^\epsilon)$ necessitates kernel feature dimension $m' = \Omega(N^{1-9\epsilon})$, which forces each attention unit to compute an $N \times N^{1-9\epsilon}$ matrix, yielding a nearly quadratic runtime. We prove Corollary 5.3 in Appendix E.3 using a similar pointer chasing reduction.

Masking-Based Sub-Quadratic Attention. Another method that reduces the computational cost of transformers is to used masked models of Λ-$\mathsf{Transformer}^N_{m,L,H}$ for a sparse mask Λ. The Longformer architecture (Beltagy et al., 2020) introduces a particular masked architecture that combines sliding windows with sparse unmasked global tokens. Put concretely, for window radius w and global frequency g, let $\Lambda^{w,g} \in \{-\infty, 0\}^{N\times N}$ be masking matrix with

$$\Lambda^{w,g}_{i,j} = \begin{cases} 0 & \text{if } |i-j| \leq w \text{ or } j \equiv 0 \pmod g, \\ -\infty & \text{otherwise.} \end{cases}$$

Then, the output of a single unit of $\Lambda^{w,g}$-masked attention is computable in time $O((w + \frac{N}{g})Nm)$.

Corollary 5.4. *Any* $T \in \Lambda^{w,g}$-$\mathsf{Attn}^N_{m,L,H}$ *with* $T(X)_N = \mathrm{hop}_k(X)_N$ *satisfies either* $L \geq k$ *or* $(w + \frac{N}{gk})mHp = \Omega(\frac{N}{k^6})$.

Like kernel-based attention, sparsely-masked attention models fail to efficiently compute hop_k. Similarly, in the same parameter-efficient regime, a Longformer must have either $w = \Omega(N^{1-9\epsilon})$ or $g = O(N^{9\epsilon})$, which jointly ensures that the masked matrix has at least $\Omega(N^{2-9\epsilon})$ entries and diminishes any computational advantages. This proof also appears in Appendix E.3.

5.4 Limitations of 1-Layer Transformers with Chain-of-Thought

While most of the paper considers transformers as sequence-to-sequence models, we can also frame them as autoregressive models performing next-token-prediction with chain-of-thought prompting. In this regime, a single causally-masked transformer aims to compute a function of its input by repeatedly predicting the next token, appending previously predicted tokens to the end of the input. In doing so, a function is computable if there exists an intermediate *chain-of-thought* produced by the model that eventually reaches the answer.

Definition 5.5. We say that $T \in \mathsf{MaskTransformer}^{N+N_{\mathrm{CoT}}}_{m,L,H}$ computes $f : \Sigma^{N+N_{\mathrm{CoT}}} \to \Sigma^N$, where the additional N tokens denote chain-of-thought, if for every $X \in \mathrm{dom}(f)$, there exists $X_{\mathrm{CoT}} \in \Sigma^{N_{\mathrm{CoT}}}$ such that $T(X \circ X_{\mathrm{CoT}})_{N:N+N_{\mathrm{CoT}}} = (X_{\mathrm{CoT}} \circ f(X))$.

The theoretical capabilities of chain-of-thought augmented transformers to simulate finite-state automata and Turing machines have been studied (Malach, 2023; Merrill & Sabharwal, 2023b), but the comparative capabilities of shallow models with chain-of-thought prompting and deep sequential models are unknown. In contrast to the fact that any transformer with N_{CoT} tokens can be simulated by a sequential model with depth scaled by N_{CoT}, we show that deep transformers cannot necessarily be efficiently simulated by shallow chain-of-thought models. We do so by demonstrating that a linear amount of chain-of-thought prompting in k is necessary to solve $\mathrm{hop}_k(X)_N$, and also sufficient.

Corollary 5.6. *Any* $T \in \mathsf{MaskTransformer}_{m,1,H}^{N+N_{\mathrm{CoT}}}$ *that computes* $\mathrm{hop}_k(X)_N$ *with* N_{CoT} *tokens of chain-of-thought requires either* $N_{\mathrm{CoT}} \geq k$ *or* $mHp = \Omega(\frac{N}{k^6})$.

The proof appears in Appendix E.4. For future work, it remains to consider the comparative powers of chain-of-thought models of depths greater than one.

6 Conclusion and Future Work

This work highlights parallelism as a central feature of transformers that sets them apart from other neural architectures. The focus on the log-depth and sublinear-width regime and specific computational tasks allows us to accentuate the benefits of parallelism, even for tasks like k-hop that appear inherently serial at first glance.

There is some efficiency loss in the "compilation" of MPC protocols to transformers that subsequent work by Sanford et al. (2024) remedies by extending Theorem 3.1 to all MPC algorithms with strictly sublinear local memory. Although we have empirically demonstrated the learnability of transformers that exploit parallelism in crucial ways, a theoretical understanding of learning such solutions remains an open question.

Finally, a unified theoretical framework for transformers and parallel computation that addresses both task parallelism and hardware paralellism would be of great value. While the MPC model delivers substantial insight into the *algorithmic capabilities* of transformers, its local memory assumptions preclude a useful analysis of the *inference and training runtime capabilities* of modern GPU hardware. Future work in this direction may integrate the results of this paper with works that present approaches to model and data parallelism (e.g. Shoeybi et al., 2019).

Acknowledgements

C.S. acknowledges the support of an NSF GRFP award and NSF grants IIS-1838154, IIS-2040971, CCF-2106429, and CCF-2211238. D.H. acknowledges the support of NSF grant IIS-2040971. The authors appreciate the literature recommendations and insights from conversations with Yuval Efron, Peilin Zhong, Alex Andoni, Rocco Servedio, and Vahab Mirrokni.

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.

References

Agarwal, A., Chapelle, O., Dudík, M., and Langford, J. A reliable effective terascale linear learning system. *Journal of Machine Learning Research*, 15(1):1111–1133, 2014.

Andoni, A., Nikolov, A., Onak, K., and Yaroslavtsev, G. Parallel algorithms for geometric graph problems. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pp. 574–583, 2014.

Andoni, A., Song, Z., Stein, C., Wang, Z., and Zhong, P. Parallel graph connectivity in log diameter rounds. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, October 2018. doi: 10.1109/focs.2018.00070. URL http://dx.doi.org/10.1109/FOCS.2018.00070.

Angluin, D., Chiang, D., and Yang, A. Masked hard-attention transformers and boolean rasp recognize exactly the star-free languages, 2023.

Assadi, S. and N, V. Graph streaming lower bounds for parameter estimation and property testing via a streaming xor lemma. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, STOC '21. ACM, June 2021. doi: 10.1145/3406325.3451110. URL http://dx.doi.org/10.1145/3406325.3451110.

Beame, P., Koutris, P., and Suciu, D. Communication steps for parallel query processing. *Journal of the ACM (JACM)*, 64(6):1–58, 2017.

Behnezhad, S., Brandt, S., Derakhshan, M., Fischer, M., Hajiaghayi, M., Karp, R. M., and Uitto, J. Massively parallel computation of matching and mis in sparse graphs. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, pp. 481–490, 2019.

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

Bengio, Y., Simard, P., and Frasconi, P. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994. doi: 10.1109/72.279181.

Bhattamishra, S., Ahuja, K., and Goyal, N. On the ability and limitations of transformers to recognize formal languages. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, 2020.

Bietti, A., Cabannes, V., Bouchacourt, D., Jegou, H., and Bottou, L. Birth of a transformer: A memory viewpoint, 2023.

Charikar, M., Ma, W., and Tan, L.-Y. New lower bounds for massively parallel computation from query complexity, 2020.

Choromanski, K., Likhosherstov, V., Dohan, D., Song, X., Gane, A., Sarlos, T., Hawkins, P., Davis, J., Mohiuddin, A., Kaiser, L., Belanger, D., Colwell, L., and Weller, A. Rethinking attention with performers, 2022.

Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

Clark, K., Khandelwal, U., Levy, O., and Manning, C. D. What does bert look at? an analysis of bert's attention. *arXiv preprint arXiv:1906.04341*, 2019.

Coy, S. and Czumaj, A. Deterministic massively parallel connectivity. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2022, pp. 162–175, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450392648. doi: 10.1145/3519935.3520055. URL https://doi.org/10.1145/3519935.3520055.

Daniely, A. Depth separation for neural networks. In Kale, S. and Shamir, O. (eds.), *Proceedings of the 2017 Conference on Learning Theory*, volume 65 of *Proceedings of Machine Learning Research*, pp. 690–696. PMLR, 07–10 Jul 2017. URL https://proceedings.mlr.press/v65/daniely17a.html.

Dao, T., Fu, D. Y., Ermon, S., Rudra, A., and Ré, C. Flashattention: Fast and memory-efficient exact attention with io-awareness. In *NeurIPS*, 2022.

Dean, J. and Ghemawat, S. Mapreduce: Simplified data processing on large clusters. In *OSDI*, pp. 137–150, 2004.

Dettmers, T., Lewis, M., Belkada, Y., and Zettlemoyer, L. Llm.int8(): 8-bit matrix multiplication for transformers at scale. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

Duris, P., Galil, Z., and Schnitger, G. Lower bounds on communication complexity. In *Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing*, pp. 81–91, 1984.

Eldan, R. and Shamir, O. The power of depth for feedforward neural networks. In Feldman, V., Rakhlin, A., and Shamir, O. (eds.), *29th Annual Conference on Learning Theory*, volume 49 of *Proceedings of Machine Learning Research*, pp. 907–940, Columbia University, New York, New York, USA, 23–26 Jun 2016. PMLR. URL https://proceedings.mlr.press/v49/eldan16.html.

Elhage, N., Nanda, N., Olsson, C., Henighan, T., Joseph, N., Mann, B., Askell, A., Bai, Y., Chen, A., Conerly, T., DasSarma, N., Drain, D., Ganguli, D., Hatfield-Dodds, Z., Hernandez, D., Jones, A., Kernion, J., Lovitt, L., Ndousse, K., Amodei, D., Brown, T., Clark, J., Kaplan, J., McCandlish, S., and Olah, C. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 2021. https://transformer-circuits.pub/2021/framework/index.html.

Ghaffari, M. Massively parallel algorithms. *URL: http://people. csail. mit. edu/ghaffari/MPA19/Notes/MPA. pdf*, 2019.

Ghaffari, M., Kuhn, F., and Uitto, J. Conditional hardness results for massively parallel computation from distributed lower bounds. In *IEEE 60th Annual Symposium on Foundations of Computer Science*, pp. 1650–1663, 11 2019. doi: 10.1109/FOCS.2019.00097.

Goodrich, M. T., Sitchinava, N., and Zhang, Q. Sorting, searching, and simulation in the mapreduce framework. In *International Symposium on Algorithms and Computation*, pp. 374–383. Springer, 2011.

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

Guha, S. and McGregor, A. Stream order and order statistics: Quantile estimation in random-order streams. *SIAM Journal on Computing*, 38(5):2044–2059, 2009. doi: 10.1137/07069328X. URL https://doi.org/10.1137/07069328X.

Hahn, M. Theoretical limitations of self-attention in neural sequence models. *Trans. Assoc. Comput. Linguistics*, 8:156–171, 2020. doi: 10.1162/tacl_{a}{_{0}{0}{3}}{0}6. URL https://doi.org/10.1162/tacl_a_00306.

Hao, Y., Angluin, D., and Frank, R. Formal language recognition by hard attention transformers: Perspectives from circuit complexity. *Trans. Assoc. Comput. Linguistics*, 10:800–810, 2022. URL https://transacl.org/ojs/index.php/tacl/article/view/3765.

Im, S., Kumar, R., Lattanzi, S., Moseley, B., Vassilvitskii, S., et al. Massively parallel computation: Algorithms and applications. *Foundations and Trends® in Optimization*, 5(4):340–417, 2023.

Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A., et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

Kacham, P., Mirrokni, V., and Zhong, P. Polysketchformer: Fast transformers via sketches for polynomial kernels, 2023.

Karloff, H., Suri, S., and Vassilvitskii, S. A model of computation for mapreduce. In *Twenty-first Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 938–948, 12 2010. doi: 10.1137/1.9781611973075.76.

Kim, J., Nguyen, T. D., Min, S., Cho, S., Lee, M., Lee, H., and Hong, S. Pure transformers are powerful graph learners, 2022.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization, 2014.

Li, Y. and McClelland, J. L. Systematic generalization and emergent structures in transformers trained on structured tasks, 2022.

Likhosherstov, V., Choromanski, K., and Weller, A. On the expressive power of self-attention matrices. *arXiv preprint arXiv:2106.03764*, 2021.

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

Loukas, A. What graph neural networks cannot learn: depth vs width. *arXiv preprint arXiv:1907.03199*, 2019.

Malach, E. Auto-regressive next-token predictors are universal learners, 2023.

Merrill, W. and Sabharwal, A. A logic for expressing log-precision transformers, 2022.

Merrill, W. and Sabharwal, A. The parallelism tradeoff: Limitations of log-precision transformers. *Transactions of the Association for Computational Linguistics*, 11: 531–545, 2023a. ISSN 2307-387X. doi: 10.1162/tacl_a_00562. URL http://dx.doi.org/10.1162/tacl_a_00562.

Merrill, W. and Sabharwal, A. The expressive power of transformers with chain of thought, 2023b.

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. ISSN 2307-387X. doi: 10.1162/tacl_a_00493. URL http://dx.doi.org/10.1162/tacl_a_00493.

MPICH. Mpi allreduce, 2023. URL https://www.mpich.org/static/docs/latest/www3/MPI_Allreduce.html.

Nanongkai, D. and Scquizzato, M. Equivalence classes and conditional hardness in massively parallel computations. *Distributed Computing*, 35(2):165–183, January 2022. ISSN 1432-0452. doi: 10.1007/s00446-021-00418-2. URL http://dx.doi.org/10.1007/s00446-021-00418-2.

Nisan, N. and Wigderson, A. Rounds in communication complexity revisited. *SIAM Journal on Computing*, 22(1): 211–219, 1993. doi: 10.1137/0222016. URL https://doi.org/10.1137/0222016.

Oren, M., Hassid, M., Adi, Y., and Schwartz, R. Transformers are multi-state rnns, 2024.

Papadimitriou, C. H. and Sipser, M. Communication complexity. In *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*, pp. 196–200, 1982.

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

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

Rogers, A., Kovaleva, O., and Rumshisky, A. A primer in bertology: What we know about how bert works. *Transactions of the Association for Computational Linguistics*, 8:842–866, 2021.

Roughgarden, T., Vassilvitskii, S., and Wang, J. Shuffles and circuits (on lower bounds for modern parallel computation). *Journal of the ACM*, 65:1–24, 11 2018. doi: 10.1145/3232536.

Sanford, C., Hsu, D., and Telgarsky, M. Representational strengths and limitations of transformers, 2023.

Sanford, C., Fatemi, B., Hall, E., Tsitsulin, A., Kazemi, M., Halcrow, J., Perozzi, B., and Mirrokni, V. Understanding transformer reasoning capabilities via graph algorithms, 2024.

Shoeybi, M., Patwary, M., Puri, R., LeGresley, P., Casper, J., and Catanzaro, B. Megatron-lm: Training multi-billion parameter language models using model parallelism, 2019.

Strobl, L. Average-hard attention transformers are constant-depth uniform threshold circuits, 2023.

Strobl, L., Merrill, W., Weiss, G., Chiang, D., and Angluin, D. Transformers as recognizers of formal languages: A survey on expressivity, 2023.

Telgarsky, M. Benefits of depth in neural networks. In Feldman, V., Rakhlin, A., and Shamir, O. (eds.), *29th Annual Conference on Learning Theory*, volume 49 of *Proceedings of Machine Learning Research*, pp. 1517–1539, Columbia University, New York, New York, USA, 23–26 Jun 2016. PMLR. URL https://proceedings.mlr.press/v49/telgarsky16.html.

Turkoglu, M. O., D'Aronco, S., Wegner, J. D., and Schindler, K. Gating revisited: Deep multi-layer rnns that can be trained. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(8):4081–4092, 2021.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. In *Advances in Neural Information Processing Systems 30*, 2017.

Wang, Z., Wang, C., Xu, X., Zhou, J., and Lu, J. Quantformer: Learning extremely low-precision vision transformers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

Wei, C., Chen, Y., and Ma, T. Statistically meaningful approximation: a case study on approximating turing machines with transformers, 2021.

Yao, S., Peng, B., Papadimitriou, C. H., 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*, 2021.

Yun, C., Bhojanapalli, S., Rawat, A. S., Reddi, S., and Kumar, S. Are transformers universal approximators of sequence-to-sequence functions? In *International Conference on Learning Representations*, 2020.

Zhang, Y., Backurs, A., Bubeck, S., Eldan, R., Gunasekar, S., and Wagner, T. Unveiling transformers with lego: a synthetic reasoning task, 2023.

A Supplemental Preliminaries

A.1 Further Details about Transformers

We discuss a few minor technicalities and modifications of the self-attention unit (Definition 2.1) and transformer model (Definition 2.2) defined in Section 2.1 that are necessary for readers looking for a comprehensive understanding of the proofs of our theoretical results.

Fixed-Bit Precision Arithmetic. As discussed in Section 2.1, we assume that all numbers that appear in the intermediate products and outputs of self-attentions are representable with p-bit precision arithmetic, where $p = \Theta(\log N)$. While the details of fixed-precision arithmetic will be uninteresting to most readers, it is necessary to explain precisely what we mean in order to ensure that proofs of results like Theorem 3.4 are sound. Throughout the paper, we allow p to depend on of constants, such as γ, δ, and ϵ.

Concretely, we assume that all query, key, and value embeddings $Q(X), K(X), V(X)$ evaluated on all inputs contain scalar values $z \in \mathbb{R}$ that are polynomially bounded (i.e. $|z| \leq \exp(O(p)) = N^{\zeta}$ for sufficiently large constant exponent $\zeta > 0$) and are inverse-polynomially discretized (i.e. $z \cdot N^{\zeta} \in \mathbb{Z}$). Depending on the desired exponent ζ, some $p = \Theta(\log N)$ can be chosen to guarantee this property. While we do not formally analyze the precision needed to approximate the particular embeddings employed by our proofs, we note that our recurring sinusoidal embeddings (e.g. Lemma D.1) can be discretized without losing their central properties and that discretizations of the restricted isometry embeddings of Proposition B.1 are analyzed by Sanford et al. (2023).

Rather than stipulating a particular bounded-precision implementation that computes the output of a self-attention unit must be implemented, we specify a rounding constraint that any computational implementation of a self-attention unit must satisfy. Precisely, we require that any output round to the same inverse-polynomial discretization as the true mathematical attention.

Definition A.1. For a self-attention unit $f \in \mathsf{Attn}_m^N$, let $\hat{f}$ be an finite-precision implementation of that unit. We say that $\hat{f}$ is a *valid implementation* if

$$\sup_{X \in \mathbb{R}^{N \times m}} \left\| f(X) - \hat{f}(X) \right\|_\infty = O\left(\frac{1}{2^p}\right).$$

This definition is only to establishing the fact that self-attention units with sufficient margins can precisely compute hardmax outputs in Lemma A.2 and to showing that MPC models can indeed compute the outputs precisely in Theorem 3.4.

Hardmax Attention. While we exclusively consider attention units with the softmax, our constructions periodically rely on the exact computation of averages of embeddings. We define the *hardmax* operator to allow the consideration of discrete averaging operations. For some $v \in \mathbb{R}^N$, let

$$\mathrm{hardmax}(X)_i = \begin{cases} \frac{1}{|I_{\max}(v)|}, & \text{if } i \in I_{\max}(v) \\ 0 & \text{otherwise,} \end{cases}$$

where $I_{\max}(v) = \{i \in [N] : v_i = \max_{i'} v_{i'}\}$.

We show that bounded-precision softmax self-attention units that satisfy a margin property can be modified slightly to have identical outputs to an analogous hardmax unit.

Lemma A.2. *Let $f \in \mathsf{Attn}_m^N$ be a self-attention unit with precision $p = \Theta(\log N)$ and embedding functions Q, K, V such that for some fixed $1 \geq \xi = N^{-O(1)}$ and every $X \in \mathbb{R}^{N \times m}$ and $i \in [N]$:*

$$A(X)_{i,i'} \leq \max_{i''} A(X)_{i,i''} - \xi, \ \forall i' \notin I_{\max}(A(X)_i),$$

where $A(X) = Q(X)K(X)^\mathsf{T}$. Then there exists a self-attention unit $f' \in \mathsf{Attn}_m^N$ with a valid p'-bit implementation with $p' = O(p)$ satisfying

$$f'(X) = \mathrm{hardmax}(A(X))V(X).$$

The proof of Lemma A.2 is provided in Appendix F.

Start Tokens. Our technical proofs are occasionally simplified by including a "dummy token" whose value is passed in self-attention layers as a default or null value. For example, in the proof of Lemma D.2, the dummy token handles the case where the reference token does not appear previously in the sequence. While we believe that this extra token is not necessary for our technical arguments, we include it for the sake of simplicity.

We model this dummy token as a *start-of-sequence* token X_0. Concretely, if we employ X_0 in a self-attention $f \in \mathsf{Attn}_m^N$ which takes as input X, we instead treat f as an attention unit in Attn_m^{N+1} that operates on $(X_0, X_1, \dots, X_N)$. We assume that X_0 is constant-valued, and therefore never both to pay attention to its outputs; it's only relevance is via its key and value embeddings $K_0(X_0), V_0(X_0) \in \mathbb{R}^m$. If X_0 is unmentioned, we assume that it does not exist, or is set such that its key embedding inner products are all zero.

Supplemental Chain-of-Thought Tokens. We periodically (see Theorem B.3 and the proofs of Corollaries 3.5 and 4.3) consider transformers with supplemental blank "chain-of-thought" tokens appended to the end of the sequence. Unlike the start token, these are only constant *at initialization* and may be used deeper in the model to perform meaningful computations.

Let $\mathsf{Transformer}_{m,L,H,d_{\text{in}},d_{\text{out}}}^{N,M}$ denote transformers with $M - N$ extra blank elements appended to the input sequence. Concretely, we represent $T \in \mathsf{Transformer}_{m,L,H,d_{\text{in}},d_{\text{out}}}^{N,M}$ as some $T' \in \mathsf{Transformer}_{m,L,H,d_{\text{in}},d_{\text{out}}}^{M}$ and define the output $T(X)$ for $X \in \mathbb{R}^{N \times d_{\text{in}}}$ by letting $Y \in \mathbb{R}^{M \times d_{\text{in}}}$ for $Y_{1:N} = X$ and $Y_{N+1:M} = \vec{0}$, and letting $T(X) = T'(Y)$.

B Proofs from Section 3.1

B.1 Proof of Lemma 3.2

Lemma 3.2. *For any $\beta, s, N \in \mathbb{N}$, there exists a transformer $\mathrm{route}_{\beta,s} \in \mathsf{Transformer}_{m,1,1}^N$ with $m = O(s^4 \beta \log N)$ satisfying $\mathrm{route}_{\beta,s}(\mathtt{Sent}) = \mathtt{Rcvd}$ for any valid (β, s)-routing $(\mathtt{Sent}, \mathtt{Rcvd})$.*

The proof relies on a *sparse propagation* sequential primitive, which complements the sparse averaging primitive of Sanford et al. (2023). For any $Q \leq d, N$, on input $X = (X_1, \dots, X_N) \in \mathbb{R}^{N \times d}$ with $X_i = (z_i, S_i) \in \mathbb{R}^{d-Q} \times [N]^Q$ and $b_i = |\{S_j \ni i : j \in [N]\}| \leq Q$, we define

$$\mathrm{sparsePropagate}_{Q,d}(X)_i = \begin{cases} \frac{1}{b_i} \sum_{S_j \ni i} z_j & \text{if } b_i > 0, \\ 0 & \text{otherwise.} \end{cases}$$

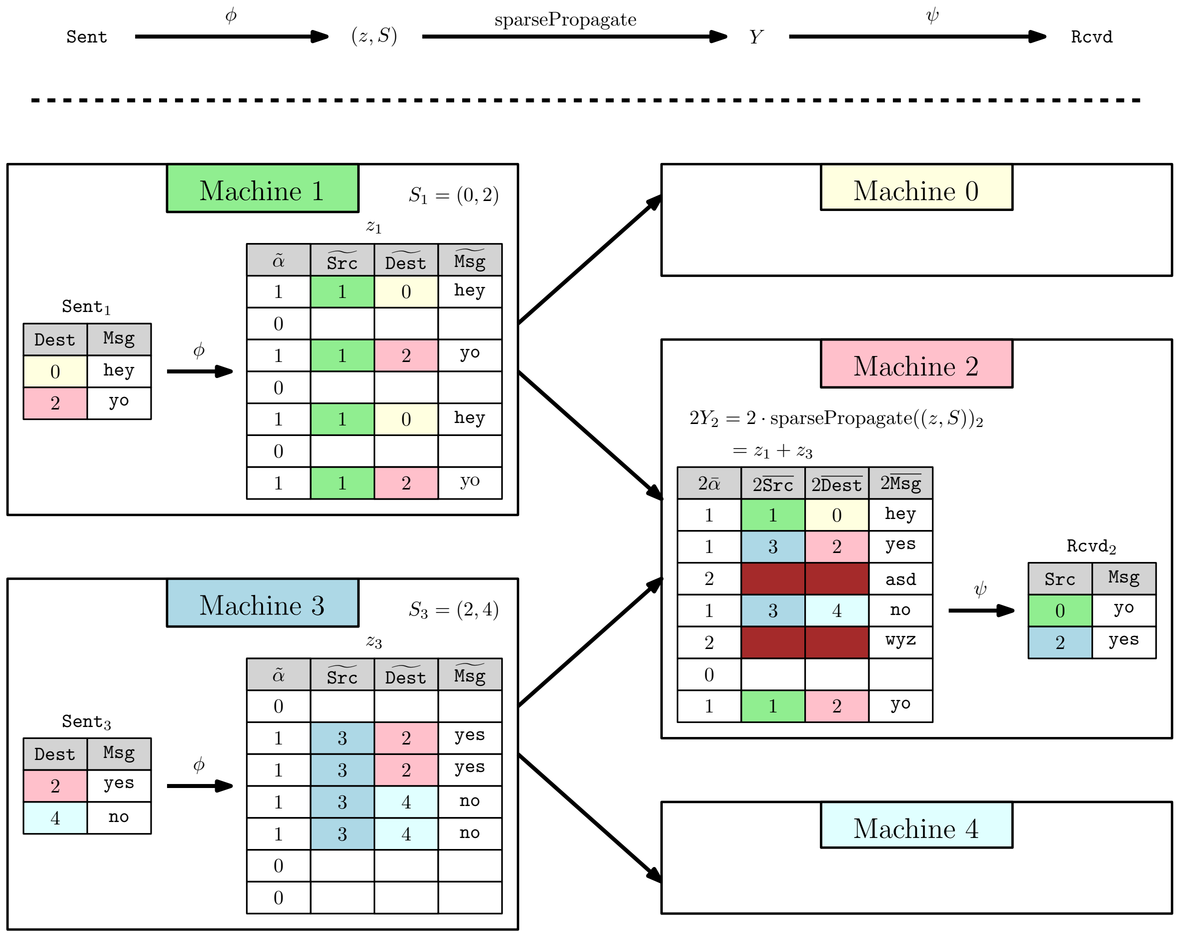
Closely following the argument of Sanford et al. (2023), we show in Proposition B.1 that there is a self-attention unit with embedding dimension $m = \max(d, O(q \log N))$ that computes $\mathrm{sparsePropagate}_{Q,d}$. This construction is a key component of the single-layer transformer used in the proof of Lemma 3.2.

Proposition B.1. *For any $b \leq N$ and d, there exists a self-attention unit $\mathrm{sparsePropagate}_{Q,d} \in \mathsf{Attn}_{m,p}^N$ for $m = d + O(Q \log N)$ and $p = O(\log N)$, which, given any input X with $X_i = (z_i, S_i, \vec{0}) \in \mathbb{R}^d \times \binom{[N]}{\leq Q} \times \{0\}^{m-Q-d}$ such that $b_i = |\{S_j \ni i : j \in [N]\}| \leq Q$ for all i, has output $\mathrm{sparsePropagate}_{Q,d}(X)$ satisfying*

$$\mathrm{sparsePropagate}_{Q,d}(X)_i = \frac{1}{b_i} \sum_{S_j \ni i} z_j.$$

The proof of Proposition B.1 appears in Appendix F.

Proof of Lemma 3.2. We construct a single-layer single-headed transformer with query, key, and value embeddings Q, K, V and output MLP ψ. Q, K, V can be decomposed as $Q = Q' \circ \phi$, $K = K' \circ \phi$, $V = V' \circ \phi$, for some input MLP ϕ and embeddings Q', K', V'. We fix Q', K', V' to be the respective embeddings of the self-attention unit with embedding dimension m from Proposition B.1 that computes $Y = \mathrm{sparsePropagate}_{s,m}(X)$ for $X_{\mathtt{Src}} = (z_{\mathtt{Src}}, S_{\mathtt{Src}})$ for every $\mathtt{Src} \in [N]$ to be determined. Hence, the proof entails designing element-wise encoders $\phi = (\phi_1, \dots, \phi_N)$ and decoders $\psi = (\psi_1, \dots, \psi_N)$ that compute $\mathtt{Rcvd}$ from $\mathtt{Sent}$, using $\mathrm{sparsePropagate}_{s,m}$ as an intermediate step. A high-level overview of the proof construction is visualized in Figure 3.



$\texttt{Sent}_1$

Dest	Msg
0	hey
2	yo

z_1

$\tilde{\alpha}$	$\widetilde{\texttt{Src}}$	$\widetilde{\texttt{Dest}}$	$\widetilde{\texttt{Msg}}$
1	1	0	hey
0			
1	1	2	yo
0			
1	1	0	hey
0			
1	1	2	yo

Machine 2

$2\tilde{\alpha}$	$2\widetilde{\texttt{Src}}$	$2\widetilde{\texttt{Dest}}$	$2\widetilde{\texttt{Msg}}$
1	1	0	hey
1	3	2	yes
2	▨	▨	asd
1	3	4	no
2	▨	▨	wyz
0			
1	1	2	yo

$\texttt{Rcvd}_2$

Src	Msg
0	yo
2	yes

$\texttt{Sent}_3$

Dest	Msg
2	yes
4	no

z_3

$\tilde{\alpha}$	$\widetilde{\texttt{Src}}$	$\widetilde{\texttt{Dest}}$	$\widetilde{\texttt{Msg}}$
0			
1	3	2	yes
1	3	2	yes
1	3	4	no
1	3	4	no
0			
0			

Figure 3. A visualization of the construction used to prove Lemma 3.2 in three phases—the encoding of each input $\texttt{Sent}_{\texttt{Src}}$ as embedding $z_{\texttt{Src}}$ and subset $S_{\texttt{Src}}$ with ϕ; the combination of those embeddings into $Y_{\texttt{Dest}}$ via the simulation of $\text{sparsePropagate}_{s,m}((z, S))$; and the decoding of each $Y_{\texttt{Dest}}$ into output $\texttt{Rcvd}_{\texttt{Dest}}$ with ψ. The figure provides an example of the encoding and decoding where machines 1 and 3 transmit messages to machine 2. "Multiple hashing" is used to compute z_1 and z_3 by encoding each message in multiple fixed-location "packets" in embedding space space. This redundancy ensures the possibility of machine 2 decoding $\texttt{Rcvd}_2$ from Y_2, due to each message occurring alone at least once in the encoding.

On input $\texttt{Sent}_{\texttt{Src}}$, we use the encodings $Q_{\texttt{Src}}, K_{\texttt{Src}}, V_{\texttt{Src}}$ to specify that all tokens $\texttt{Dest}$ with $\texttt{Dest} \in \texttt{Sent}_{\texttt{Src}}$ (or equivalently, all $\texttt{Dest}$ with $\texttt{Src} \in \texttt{Rcvd}_{\texttt{Dest}}$) should receive a copy of the encoding of $\texttt{Sent}_{\texttt{Src}}$. That is, we set $S_{\texttt{Src}} := \{\texttt{Dest} \in \texttt{Sent}_{\texttt{Src}}\}$ for each $\texttt{Src} \in [N]$. This ensures that Y satisfies

$$Y_{\texttt{Dest}} = \frac{1}{|\texttt{Rcvd}_{\texttt{Dest}}|} \sum_{\texttt{Src} \in \texttt{Rcvd}_{\texttt{Dest}}} z_{\texttt{Src}}.$$

While it's tempting to simply set each $z_{\texttt{Src}} \in \mathbb{R}^m$ equal to a (βs)-dimensional vectorization of $\texttt{Sent}_{\texttt{Src}}$, it is unclear how to extract $\texttt{Rcvd}_{\texttt{Dest}}$ from each $Y_{\texttt{Dest}}$, since each average performed by $\text{sparsePropagate}_{s,m}$ will combine multiple vector embeddings in a shared space. In order to avoid these troubles, we employ a *multiple hasing-based encoding* that treats messages as "packets" identified by a message, a source, a destination, and a "validity token" that can be used to determine whether a message is uncorrupted. We include multiple copies of each packet in the encoding $z_{\texttt{Src}}$. For notational ease, we represent each $z_{\texttt{Src}} \in \mathbb{R}^m$ as a collection of packets

$$z_{\texttt{Src}} = (\widetilde{\texttt{Msg}}_{\texttt{Src},j}, \widetilde{\texttt{Src}}_{\texttt{Src},j}, \widetilde{\texttt{Dest}}_{\texttt{Src},j}, \alpha_{\texttt{Src},j})_{j \in [m']} \in (\mathbb{Z}_{2^p}^{\beta} \times [N] \times [N] \times \{0, 1\})^{m'},$$

where $m = m'(3 + \beta)$.

To sparsely and redundantly encode each $\mathtt{Sent}_{\mathtt{Src}}$ as $z_{\mathtt{Src}}$, we encode outgoing messages as packets by utilizing the matrix A guaranteed by the following fact (which we use with $n := N^2$, $b := s^2$, and $m' := d = O(s^4 \log N)$).

Fact B.2. *For any n, $b \leq n$, and $d \geq \lceil 12b^2 \ln n \rceil$, there exists a binary matrix $A \in \{0,1\}^{n \times d}$ such that, for every subset $S \subseteq [n]$ with $|S| \leq b$, the columns of the sub-matrix $A_S \in \{0,1\}^{|S| \times d}$ contains all S-dimensional elementary vectors, i.e., $\{e_1, \dots, e_{|S|}\}$ is a subset of the columns of A_S.*

The proof of Fact B.2 is at the end of the section. We use the following rule to determine which (if any) message to encode as a packet at each $\mathtt{Src} \in [N]$ and $j \in [m']$. We let $A_{(\mathtt{Src},\mathtt{Dest}),j} = A_{N(\mathtt{Src}-1)+\mathtt{Dest},j}$ for notational convenience.

$$z_{\mathtt{Src},j} = \begin{cases} (\mathtt{Msg}, \mathtt{Src}, \mathtt{Dest}, 1) & \text{if } (\mathtt{Msg}, \mathtt{Dest}) \in \mathtt{Sent}_{\mathtt{Src}} \text{ and } A_{(\mathtt{Src},\mathtt{Dest}),j} = 1 \\ & \quad \text{and } A_{(\mathtt{Src},\mathtt{Dest}'),j} = 0,\ \forall\, \mathtt{Dest}' \in \mathtt{Sent}_{\mathtt{Src}} \setminus \{\mathtt{Dest}\}, \\ (\vec{0}, 0, 0, 0) & \text{otherwise.} \end{cases}$$

In Figure 3, this encoding is visualized in the tables of "Machine 1" and "Machine 3," where the entirety of each message is encoded in two fixed and distinct locations in the embeddings z_1 and z_3, alongside metadata about the source of message and the validity $\tilde{\alpha}$. Each message is encoded as multiple identical packets in different embedding dimensions and a large fraction of embedding locations are left blank. These features are critical for the proper evaluation of the decoding step ψ.

We analyze the $Y = \text{sparsePropagate}_{\beta,m}(X)$ outputs, letting

$$Y_{\mathtt{Dest}} = (Y_{\mathtt{Dest},1}, \dots, Y_{\mathtt{Dest},m'}), \quad Y_{\mathtt{Dest},j} \in (\mathbb{R}^\beta \times \mathbb{R} \times \mathbb{R} \times \mathbb{R})^{m'},$$

with all numbers represented with p-bit fixed precision. This analysis shows that there exists an element-wise decoder MLP ψ satisfying $\psi_{\mathtt{Dest}}(Y_{\mathtt{Dest}}) = \mathtt{Rcvd}_{\mathtt{Dest}}$ for all $\mathtt{Dest} \in [N]$. For any $j \in [m']$, observe from the definition of $z_{\mathtt{Src}}$ and $\text{sparsePropagate}_{s,m}$ that

$$\begin{aligned} Y_{\mathtt{Dest},j} &=: \left(\overline{\mathtt{Msg}}_{\mathtt{Dest},j}, \overline{\mathtt{Src}}_{\mathtt{Dest},j}, \overline{\mathtt{Dest}}_{\mathtt{Dest},j}, \bar{\alpha}_{\mathtt{Dest},j}\right) \\ &= \frac{1}{|\mathtt{Rcvd}_{\mathtt{Dest}}|} \sum_{\mathtt{Src} \in \mathtt{Rcvd}_{\mathtt{Dest}}} \left(\widetilde{\mathtt{Msg}}_{\mathtt{Src},j}, \widetilde{\mathtt{Src}}_{\mathtt{Src},j}, \widetilde{\mathtt{Dest}}_{\mathtt{Src},j}, \alpha_{\mathtt{Src},j}\right). \end{aligned}$$

Before formally analyzing this construction, we motivate its utility with Figure 3. The encoding $2Y_2$ of Machine 2 contains four "clean" rows j with $2\bar{\alpha}_{2,j} = 1$, two "corrupted" rows with $2\bar{\alpha}_{2,j} = 2$, and one "blank" row with $2\bar{\alpha}_{2,j} = 0$.

- The **blank row** contains no information about any incoming messages, since neither Machine 1 nor Machine 3 encoded messages as packets in these locations. The fact that $2\bar{\alpha}_{2,j} = 0$ certifies the blankness of this row, and hence, the decoder ψ can ignore it.
- The **corrupted rows** correspond to locations where both Machine 1 and Machine 3 saved messages as packets. As a result, the corresponding embedding $Y_{2,j} = \frac{1}{2}(z_{1,j} + z_{3,j})$ is an average of two non-zero embeddings and is hence "corrupted." Because $2\bar{\alpha}_{2,j} = 2$, the decoder ψ detects the corruption and ignores it when computing $\mathtt{Rcvd}_2$.
- The **clean rows** are locations where exactly one of Machine 1 and Machine 3 encoded a message. Hence, these messages can be cleanly understood by the decoder ψ, which simply validates the "cleanliness" of the row with $2\bar{\alpha}_{2,j} = 1$, determines whether Machine 2 is indeed the target recipient of the respective message, and saves all such messages in the decoding $\mathtt{Rcvd}_2$.

We prove the validity of this intuition by ensuring that the encoding scheme successfully encodes each incoming message in a clean row and that the category of each row (blank, corrupted, or clean) can be detected by the decoder ψ. We observe the following sequence of facts about every $Y_{\mathtt{Dest}}$. Let

$$\mathtt{Relevant}_{\mathtt{Dest}} := \{(\mathtt{Msg}, \mathtt{Src}', \mathtt{Dest}') : \mathtt{Src}' \in \mathtt{Rcvd}_{\mathtt{Dest}},\ (\mathtt{Msg}, \mathtt{Dest}') \in \mathtt{Sent}_{\mathtt{Src}'}\}$$

denote the set of *all* messages sent by sources of messages sent to $\mathtt{Dest}$.

1. Consider any outgoing message $(\texttt{Msg}, \texttt{Src}', \texttt{Dest}') \in \texttt{Relevant}_{\texttt{Dest}}$. By the property of A guaranteed by Fact B.2, there exists some j such that $A_{(\texttt{Src}', \texttt{Dest}'),j} = 1$ and $A_{(\texttt{Src}'', \texttt{Dest}''),j} = 0$ for every $(\texttt{Src}'', \texttt{Dest}'') \in \texttt{Relevant}_{\texttt{Dest}} \setminus \{(\texttt{Src}', \texttt{Dest}')\}$. As a result of the definition of the encoding z and the averaged representation of $Y_{\texttt{Dest}}$:

$$Y_{\texttt{Dest},j} = \frac{1}{|\texttt{Rcvd}_{\texttt{Dest}}|} (\texttt{Msg}, \texttt{Src}', \texttt{Dest}', 1) \, . \tag{1}$$

2. Conversely, if $\bar{\alpha}_{\texttt{Dest},j} = 1/|\texttt{Rcvd}_{\texttt{Dest}}|$, then there exists a unique $(\texttt{Msg}, \texttt{Src}', \texttt{Dest}') \in \texttt{Relevant}_{\texttt{Dest}}$ such that (1) is satisfied.

3. If at least one message is received, then the minimal nonzero value of $\bar{\alpha}_{\texttt{Dest}}$ is $1/|\texttt{Rcvd}_{\texttt{Dest}}|$.

We design $\psi_{\texttt{Dest}}$ to uniquely identify $\texttt{Rcvd}_{\texttt{Dest}}$ from $Y_{\texttt{Dest}}$ as follows. If at least one message is received, then $1/|\texttt{Rcvd}_{\texttt{Dest}}|$ can be identified by finding the smallest nonzero value of $\bar{\alpha}_{\texttt{Dest}}$. The decoder ψ inspects every $Y_{\texttt{Dest},j}$ satisfying $\bar{\alpha}_{\texttt{Dest},j} = 1/|\texttt{Rcvd}_{\texttt{Dest}}|$, which therefore satisfies

$$|\texttt{Rcvd}_{\texttt{Dest}}| \cdot (\overline{\texttt{Msg}}_{\texttt{Dest},j}, \overline{\texttt{Src}}_{\texttt{Dest},j}, \overline{\texttt{Dest}}_{\texttt{Dest},j}) \in \texttt{Relevant}_{\texttt{Dest}}.$$

Thus, if $|\texttt{Rcvd}_{\texttt{Dest}}| \cdot \overline{\texttt{Dest}}_{\texttt{Dest},j} = \texttt{Dest}$, then $|\texttt{Rcvd}_{\texttt{Dest}}| \cdot (\overline{\texttt{Msg}}_{\texttt{Dest},j}, \overline{\texttt{Src}}_{\texttt{Dest},j}) \in \texttt{Rcvd}_{\texttt{Dest}}$, and ψ encodes it as such.

□

Fact B.2. *For any n, $b \leq n$, and $d \geq \lceil 12b^2 \ln n \rceil$, there exists a binary matrix $A \in \{0,1\}^{n \times d}$ such that, for every subset $S \subseteq [n]$ with $|S| \leq b$, the columns of the sub-matrix $A_S \in \{0,1\}^{|S| \times d}$ contains all S-dimensional elementary vectors, i.e., $\left\{e_1, \dots, e_{|S|}\right\}$ is a subset of the columns of A_S.*

Proof. Let $\mathrm{col}(A)$ denote the set of columns of A. We use the probabilistic method and consider A with iid entries $A_{i,j} \sim \mathrm{Bernoulli}(\frac{1}{b+1})$. We bound the probability of failure:

$$\begin{aligned}
\Pr\left[\exists S \in \binom{[n]}{\leq b} \text{ s.t. } \left\{e_1, \dots, e_{|S|}\right\} \not\subset \mathrm{col}(A_S)\right] &\leq b \cdot n^b \Pr\left[e_i \notin \mathrm{col}(A_S)\right] \\
&\leq n^{b+1} \left(1 - \frac{1}{b+1} \cdot \left(1 - \frac{1}{b+1}\right)^b\right)^d \\
&\leq n^{b+1} \left(1 - \frac{1}{e(b+1)}\right)^d \\
&\leq n^{b+1} \cdot \exp\left(-\frac{d}{e(b+1)}\right) \\
&< \exp\left((b+1)\ln n - \frac{d}{3(b+1)}\right) \leq 1.
\end{aligned}$$

Therefore, there exists a matrix A with the claimed property. □

B.2 Proof of Theorem 3.1

We give a generalization of Theorem 3.1 that simulates a broader family of MPC protocol, including those with more than n machines (i.e. $\gamma \geq \delta$). We accommodate this generalization by simulating MPC protocols with the generalized transformer family $\mathsf{Transformer}_{m,L,H}^{N,M}$ detailed in Appendix A with supplemental blank "chain-of-thought" tokens.

Theorem B.3 (Generalization of Theorem 3.1)**.** *For constant $\gamma, \delta > 0$ and any potentially randomized R-round (γ, δ)-MPC protocol π on n_{in} input words and $n_{\mathrm{out}} \leq n_{\mathrm{in}}$ output words, there exists a transformer $T \in \mathsf{Transformer}_{m,L,H}^{N,M}$ with $N = n_{\mathrm{in}}, M = \max(n_{\mathrm{in}}, O(n_{\mathrm{in}}^{1+\gamma-\delta})), m = O(n_{\mathrm{in}}^{4\delta} \log n_{\mathrm{in}}), L = R+1, H = O(\log\log n_{\mathrm{in}})$ such that*

$$T(\texttt{Input})_{:n_{\mathrm{out}}} = \pi(\texttt{Input}).$$

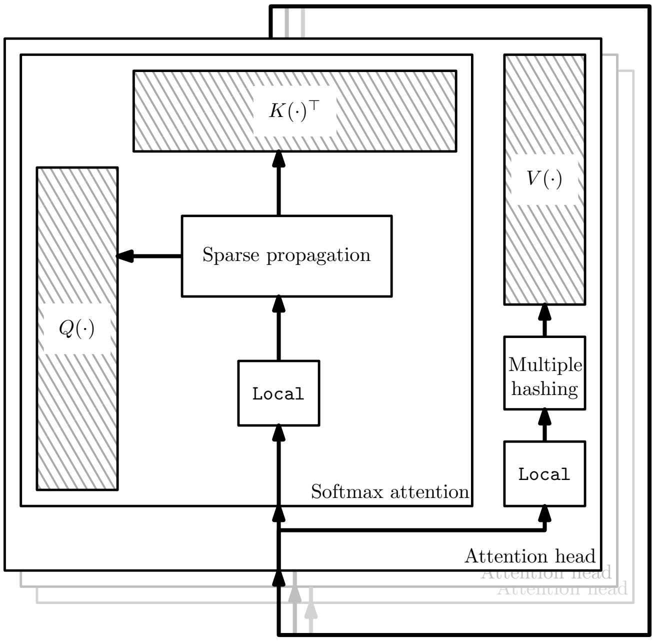


Figure 4. To simulate MPC, the local computation within each machine is pushed inside $Q(\cdot), K(\cdot), V(\cdot)$, and then the pairwise attention matrix performs message routing. To ensure proper routing and also that the outputs of $Q(\cdot), K(\cdot), V(\cdot)$ are all tall-and-skinny matrices, the construction carefully utilizes both multiple hashing and sparse propagation.

Theorem 3.1 is an immediate consequence of Theorem B.3 by noting that $M = N$ for sufficiently large n_{in} when $\gamma < \delta$. Its central construction is summarized in Figure 4.

Proof. Consider any MPC protocol π with $q = O(n_{\text{in}}^{1+\gamma-\delta})$ machines and $s = O(n_{\text{in}}^{\delta})$ local memory that, following the notation of Definition 2.3, maps $\texttt{Input} \in \mathbb{Z}_{2^p}^{n_{\text{in}}}$ to $\texttt{Output} \in \mathbb{Z}_{2^p}^{n_{\text{out}}}$ with intermediates $\texttt{MachineIn}^{(1)}, \dots \texttt{MachineIn}^{(R)}$ and $\texttt{MachineOut}^{(1)}, \dots, \texttt{MachineOut}^{(R)}$ and deterministic functions $(\texttt{Local}_{r,i})_{r \in [R], i \in [q]}$ with

$$\texttt{MachineOut}_i^{(r)} = \texttt{Local}_{r,i}(\texttt{MachineIn}_i^{(r)}).$$

To simulate the protocol, we let every machine $i \in [q]$ correspond to a particular position in the transformer's context. A transformer that simulates π can then be constructed that consolidates $\texttt{Input}$ onto $\lceil n_{\text{in}}/s \rceil$ machines to match $\texttt{MachineIn}^{(1)}$; computes $\texttt{MachineIn}^{(r+1)}$ from $\texttt{MachineIn}^{(r)}$ for each $r = 1, \dots, R-1$; and computes and properly distributes $\texttt{Output}$ from $\texttt{MachineIn}^{(r)}$. These three elements of the construction exist due to the following lemmas, which are proved later.

Lemma B.4. *For any MPC protocol π with local memory s and q machines with n_{in}-word inputs, there exists a transformer* $\text{init} \in \mathsf{Transformer}_{s,1,1,d_{\text{in}},d_{\text{out}}}^{n_{\text{in}},\max(n_{\text{in}},q)}$ *with $d_{\text{in}} = 1$ and $d_{\text{out}} = s$, which, given* $\texttt{Input} \in \mathbb{Z}_{2^p}^{n}$*, has output satisfying* $\text{init}(\texttt{Input}) = \texttt{MachineIn}^{(1)}$.

Lemma B.5. *For any R-round MPC protocol π with local memory s and q machines and any $r \in [R-1]$, there exists a transformer* $\text{round}^{(r)} \in \mathsf{Transformer}_{m,1,H,d_{\text{in}},d_{\text{out}}}^{q}$ *with $H = O(\log\log q)$, $m = O(s^4 \log q)$, and $d_{\text{in}} = d_{\text{out}} = s$ which, given any valid input $X = \texttt{MachineIn}^{(r)} \in \mathbb{Z}_{2^p}^{q \times m}$ under the MPC protocol in vectorized form, has output satisfying* $\text{round}^{(r)}(X) = \texttt{MachineIn}^{(r+1)}$.

Lemma B.6. *For any R-round MPC protocol π with local memory s and q machines with n_{out}-word output, there exists a transformer* $\text{final} \in \mathsf{Transformer}_{s,1,1,d_{\text{in}},d_{\text{out}}}^{q,\max(n_{\text{out}},q)}$ *for $d_{\text{in}} = s$ and $d_{\text{out}} = 1$, which, given input $X = \texttt{MachineIn}^{(R)}$, has output* $\text{final}(X)$ *with* $\text{final}(X)_{i,1} = \texttt{Output}_i \in \mathbb{Z}_{2^p}$.

The proof immediate from the three lemmas. We construct the final transformer T by stacking the single-layer constructions as a single transformer with embedding dimension m:

$$T = \mathrm{final} \circ \mathrm{round}^{(R-1)} \circ \cdots \circ \mathrm{round}^{(1)} \circ \mathrm{init}.$$

The proofs of Lemmas B.4 and B.6 rely on simple constructions with fixed attention matrices and appear in Appendix F. The proof of Lemma B.5 relies on Lemma 3.2 and is proved in the following section. □

Proof of $\mathrm{round}^{(r)}$ Construction. To prove the existence single-layer transformer that simulates $\mathrm{round}^{(r)}$, we separate the computational task into two steps: (i) obtaining $\texttt{MachineOut}^{(r)}$ from $\texttt{MachineIn}^{(r)}$ and (ii) obtaining $\texttt{MachineIn}^{(r+1)}$ from $\texttt{MachineOut}^{(r)}$. Because the former requires no communication between machines, we can encode that conversion in the input MLP to the transformer.

The nontrivial part of the reduction is thus the latter step, which we obtain by utilizing multiple single-headed attention units $\mathrm{route}_{\beta,s}$ of Lemma 3.2 to route messages of different sizes to their recipients. The difficulty in this task is the mismatch in functionality between the two computational models: while the MPC model ensures that each recipient automatically receives its intended messages, transformers must implement this functionality manually, while ensuring that multiple messages do not overwrite one another.

The following lemma implements that routing functionality for all messages, using different attention heads depending on the size of the message. We prove Lemma B.5 at the end of the section as a simple modification of Lemma B.7.

Lemma B.7. *For any R-round MPC protocol π with local memory s and q machines and any $r \in [R-1]$, there exists a transformer $\mathrm{route}^{(r)} \in \mathsf{Transformer}^q_{m,1,H}$ with $H = O(\log\log q)$ and $m = O(s^4 \log q)$, which, given any valid input $X = \texttt{MachineOut}^{(r)} \in \mathbb{Z}_{2^p}^{q\times m}$ under the MPC protocol in vectorized form, has output satisfying $\mathrm{route}^{(r)}(X) = \texttt{MachineIn}^{(r+1)}$.*

Because at most s messages can be shared and received by each machine, and each message is of size at most s, we can prove an single-headed alternative to Lemma B.7 with a somewhat suboptimal dependence on embedding dimension. By applying by Lemma 3.2 with message size $\beta = s$, bounded number of messages s, and context length $N = q$, there exists a transformer $\mathrm{route}_{s,s}$ with $H = 1$ and $m = O(s^5 \log q)$ that computes $\texttt{MachineIn}^{(r+1)}$ from $\texttt{MachineOut}^{(r+1)}$ by regarding each outgoing message as belonging to $\mathbb{Z}_{2^p}^s$ by adding padding dimensions as needed.

We improve the embedding dimension to $m = O(s^4 \log q)$ by running in parallel $O(\log\log N)$ transformers guaranteed by Lemma 3.2 that encode differently sized messages. The number of heads H increases at a doubly-logarithmic rate because of a doubling trick employed on the size of message encodings used by constituent part.

Proof. We describe an implementation of $\mathrm{route}^{(r)}$ by considering any fixed input $\texttt{MachineOut}^{(r)} \in \mathbb{Z}_{2^p}^{q\times m}$. For each $i \in [q]$ and some integer sequence $1 = \beta_0 < \beta_1 < \cdots < \beta_H = s+1$, we partition $\texttt{MachineOut}_i^{(r)}$ into H disjoint subsets as follows. For any $h \in [H]$, let

$$\begin{aligned}\texttt{Sent}_i^h &:= \left\{(\texttt{Msg}, \texttt{Dest}) \in \texttt{MachineOut}_i^{(r)} : \dim(\texttt{Msg}) \in [\beta_{h-1}, \beta_h]\right\},\\ \texttt{Rcvd}_i^h &:= \left\{(\texttt{Msg}, \texttt{Src}) \in \texttt{MachineIn}_i^{(r+1)} : \dim(\texttt{Msg}) \in [\beta_{h-1}, \beta_h]\right\},\end{aligned}$$

and note that $\texttt{MachineOut}_i^{(r)} = \dot{\bigcup}_{h=1}^H \texttt{Sent}_i^h$ and $\texttt{MachineIn}_i^{(r+1)} = \dot{\bigcup}_{h=1}^H \texttt{Rcvd}_i^h$.

For each $h \in [H]$, note that $\dim(\texttt{Msg}) \le \beta_h$, and $\left|\texttt{Sent}_i^h\right| = \left|\texttt{Rcvd}_i^h\right| \le s/\beta_{h-1}$. As a result, Lemma 3.2 guarantees the existence of a single-headed transformer $\mathrm{route}_h^{(r)}$ such that $\mathrm{route}_h^{(r)}(\texttt{Sent}^h) = \texttt{Rcvd}^h)$ with embedding dimension $m_h \le Cs^4\beta_h \log(q)/\beta_{h-1}^4$ for some sufficiently large universal constant C.

We defined $\mathrm{route}^{(r)}$ as the computation of $\mathrm{route}_1^{(r)}, \dots, \mathrm{route}_H^{(r)}$ as H parallel heads of self-attention with disjoint embeddings concatenated into in m-dimensional embedding space with $m = \sum_{h=1}^H m_h$. We conclude by letting

$$\beta_h = \begin{cases} 1 & \text{if } h = 0, \\ \min(2\beta_{h-1}^3, q+1) & \text{if } h \in [H], \end{cases}$$

noting that $\beta_H = q+1$ for $H = O(\log\log q)$, and bounding m:

$$m \le \sum_{h=1}^{H} \frac{Cs^4 \log(q)\beta_h}{\beta_{h-1}^4} \le 2Cs^4 \log(q) \cdot \sum_{h=1}^{H} \frac{1}{\beta_{h-1}}$$

$$\le 2Cs^4 \log(q) \cdot \sum_{h=1}^{H} \frac{1}{2^{h-1}} = O(s^4 \log q). \qquad \square$$

Proof of Lemma B.5. To simulate a round of MPC protocol π by mapping $\texttt{MachineIn}^{(r)}$ and ρ_r to $\texttt{MachineIn}^{(r+1)}$, the single-layer transformer $\mathrm{round}^{(r)}$ first computes $\texttt{MachineOut}^{(r)}$ element-wise and then properly routes messages in $\texttt{MachineOut}^{(r)}$ to their proper destination. We can define $\mathrm{round}^{(r)} = \mathrm{route}^{(r)} \circ \texttt{Local}_r$ for $\mathrm{route}^{(r)}$ in Lemma B.7 and $\texttt{Local}_{r,i}(\texttt{MachineIn}_i^{(r)}, \rho_{r,i}) = \texttt{MachineOut}_i^{(r)}$. This can be immediately constructed as a single-layer transformer by prepending the embeddings Q, K, V of the construction of $\mathrm{route}^{(r)}$ with $\texttt{Local}_r$, using $Q \circ \texttt{Local}_r$, $K \circ \texttt{Local}_r$, $V \circ \texttt{Local}_r$ as the embeddings of $\mathrm{round}^{(r)}$. □

B.3 Additional Graph Problems Solvable by Log-Depth Transformers

Theorem 8.1 and Corollary 8.2 of Coy & Czumaj (2022) give efficient MPC protocols for other graph problems besides connectivity, and therefore, as corollaries of Theorem 3.1, we also obtain log-depth transformers for these problems.

Corollary B.8 (Spanning forest construction)**.** *For any constant $\epsilon \in (0,1)$ and any $D \le N$, there exists a transformer in $\mathsf{Transformer}^N_{m,L,H}$ with $m = O(N^\epsilon)$, $H = O(\log\log N)$, and $L = O(\log D)$ that computes a rooted spanning forest of any input graph $G = (V,E)$ with $|V|, |E| = O(N)$ where each connected component has diameter at most D.*

Corollary B.9 (Minimum spanning forest construction)**.** *For any constant $\epsilon \in (0,1)$ and any $D_{MSF} \le N$, there exists a transformer in $\mathsf{Transformer}^N_{m,L,H}$ with $m = O(N^\epsilon)$, $H = O(\log\log N)$, and $L = O(\log D_{MSF})$ that identifies the connected components of any input graph $G = (V,E)$ with $|V|, |E| = O(N)$ and $\mathrm{poly}(N)$-bounded integer weights whose minimum spanning forest has diameter at most D_{MSF}.*

C Proofs from Section 3.2

C.1 Proof of Theorem 3.4

As in Appendix B.2, we give and prove a generalized version of Theorem 3.4 that broadens the family of considered transformers to include masked models and those that contain extra blank chain-of-thought tokens, using notation from Appendix A.

Theorem C.1 (Generalization of Theorem 3.4)**.** *For any transformer $T \in \mathsf{Transformer}^{N,M}_{m,L,H}$ (or $\mathsf{MaskTransformer}^{N,M}_{m,L,H}$) with $mH = O(N^\delta)$ for $\delta \in (0,1)$ and $M = \Theta(N^{1+\alpha})$ for $\alpha \ge 0$ and for any $\delta' \in (\delta, 1)$, there exists an $O(\frac{L(1+\alpha)}{\delta'-\delta})$-round $(1+2\alpha+\delta', \delta')$-MPC protocol with $q = O(M^2)$ machines with $s = O(N^{\delta'})$ local memory that outputs the same sequence as $T(X)$ for all $X \in \mathbb{R}^N$.*

Theorem 3.4 is an immediate consequence by setting $M := N$ and $\alpha := 0$.

Proof. It suffices to show that an $O(\frac{1+\alpha}{\delta'-\delta})$-round MPC protocol π that simulates a single-layer transformer $T \in \mathsf{Transformer}^M_{m,m,m,1,H}$ with m-dimensional input and output embeddings since a depth-L transformer can be constructed by applying L such protocols sequentially. Moreover, we can ignore the difference between the input context length N and the context length with padding M by assuming that the input contains M tokens.

Concretely, we consider H heads with embeddings $(Q_h, K_h, V_h)_{h\in[H]}$, element-wise output MLP $\psi = (\psi_1, \dots, \psi_M)$, and any fixed masks $\Lambda_1, \dots, \Lambda_H \in \{-\infty, 0\}^{M\times M}$. We show that there exists some π such that for any $\texttt{Input} = X \in \mathbb{R}^{M\times m}$,

$$\pi(X) = \psi\left(X + \sum_{h=1}^{H} \mathrm{softmax}(Q_h(X)K_h(X)^\mathsf{T} + \Lambda_h)V_h(X)\right),$$

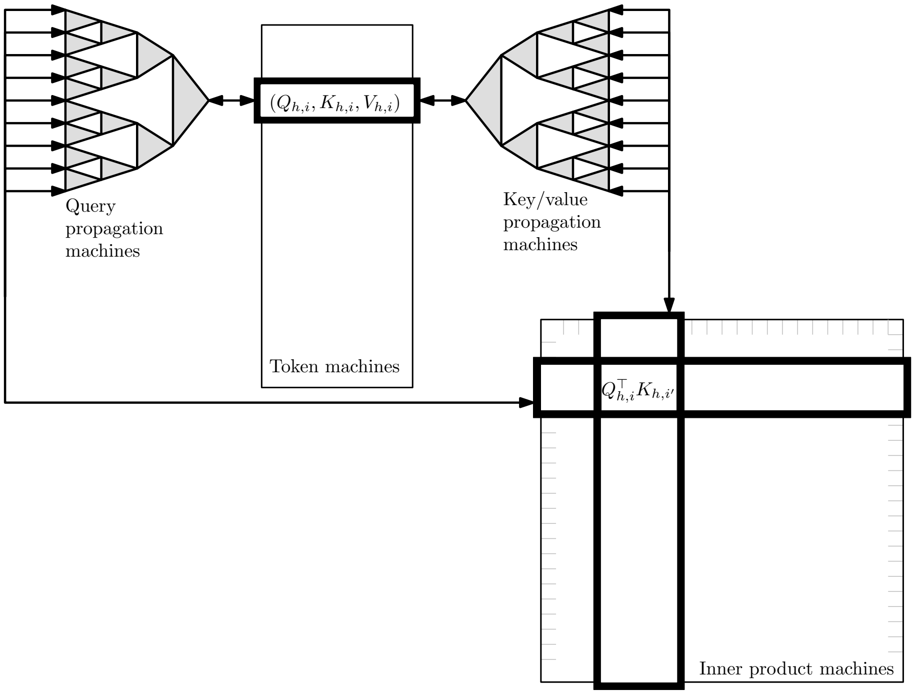

$(Q_{h,i}, K_{h,i}, V_{h,i})$
Query propagation machines
Key/value propagation machines
Token machines
$Q_{h,i}^\top K_{h,i'}$
Inner product machines

Figure 5. This construction employs M^2 *inner product machines* to compute the entries of the softmax matrix, and M *token machines* to compute all values of $Q(\cdot), K(\cdot), V(\cdot)$. What is most complex about the construction are the additional machines and message routing needed to propagate these values efficiently between the inner product machines and the token machines, in particular carefully aggregating the output of the attention mechanism and computing its normalization. To this end, the protocol uses additional machines, organized into a tree with branching factor $b = O(N^{\delta' - \delta})$ and depth $D = O(\frac{1+\alpha}{\delta' - \delta})$.

where numbers in X and all intermediate products of the transformer computation can be represented with $p = O(\log M)$ bit precision.

Our MPC protocol π, which will use $q = O(M^2)$ machines and $s = \Theta(N^{\delta'})$ words of local memory per machine, assigns each of the q machines to one of four possible roles: token machine, inner product machine, query propagation machine, and key/value propagation machine. We describe these machines below. For the sake of readability, we identify machines with easily interpretable descriptions and use the bijection $\texttt{ID}$ to map each of those to a token in $[q]$ that is used for routing messages. Our protocol has two important parameters: $b = \lfloor s/(4mH) \rfloor = O(N^{\delta' - \delta})$ is the *branching factor* of the protocol, and $D = \lceil \log_b(M) \rceil = O(\frac{1+\alpha}{\delta' - \delta})$ is the *depth* of the protocol.

At a high level (see Figure 5 for a corresponding diagram), the protocol involves computing all intermediate products of the of a transformer unit by performing MLP computations in N *token machines*, computing inner products in N^2 *inner product machines*, and using $O(N^2)$ other *propagation machines* arranged in trees to share information between the two in $O(D)$ rounds. The protocol draws inspiration from Appendix C.6.1 of Sanford et al. (2023), which uses a similar construction to simulate transformers with CONGEST protocols on fixed graphs. It is also similar to the MPC implementation of the MPI AllReduce functionality (MPICH, 2023) described by Agarwal et al. (2014).

- Machine $i \in [M]$ is a *token machine* that performs all element-wise computation on the ith token embedding, including the computation of $(Q_{h,i}(X_i), K_{h,i}(X_i), V_{h,i}(X_i))_{h \in [H]}$ and the final MLP output ψ_i. Let $\texttt{ID}(i) = i$.
- Machine $(i, i') \in [M]^2$ is an *inner product machine* designed to compute the inner products $(Q_{h,i}(X_i)^\mathsf{T} K_{h,i'}(X_{i'}))_{h \in [H]}$.
- Machine $(\texttt{Q}, i, d, k)$ for token $i \in [M]$, depth $d \in [D-1]$ and position $k \in [b^d]$ is a *query propagation machine*. This machine is responsible for handling communication of query tokens $(Q_{h,i}(X_i))_{h \in [H]}$ and of all partially-computed attention outputs for the ith token between token machine i and inner product machines (i, i') for

$$i' \in \texttt{Descendants}_{d,k} := \left\{b^{D-d}(k-1), \dots, b^{D-d}k\right\} \cap [M].$$

Concretely, if $\ell = 1$, then the machine communicates with token machine i and query propagation machines $(\mathtt{Q}, i, d+1, k')$ for

$$k' \in \mathtt{Children}_k := \{b(k-1)+1, \dots, bk\}\,.$$

If $\ell = D-1$, then it communicates with inner product machines (i, i') for $i' \in \mathtt{Children}_k \cap [M]$ and query propagation machine $(\mathtt{Q}, i, d-1, \lfloor k/b \rfloor)$. Otherwise, it communicates with query propagation machines $(\mathtt{Q}, i, d-1, \mathtt{Parent}_k)$, for $\mathtt{Parent}_k := \lfloor k/b \rfloor$, and $(\mathtt{Q}, i, d+1, k')$ for $k' \in \mathtt{Children}_k$.

- Machine $(\mathtt{KV}, i, d, k)$ is a *key/value propagation machine*. This machine is analogous to a query propagation machine, except that it is responsible for the communication of key and value tokens $(Q_{h,i}(X_i), V_{h,i}(X_i))_{h\in[H]}$ between token machine i and inner product machines (i, i') for $i' \in \mathtt{Descendants}_{d,k}$.

Since the total number of machines is $q = M + M^2 + M\sum_{d=1}^{D-1} b^d = O(M^2)$, we conclude that the global memory of the protocol is $qs = O(N^{2+2\alpha+\delta'})$, which means the protocol is $(1+2\alpha+\delta', \delta')$-MPC. We simulate the transformer using a four stage protocol using $2D+3 = O(\frac{1+\alpha}{\delta'-\delta})$ rounds of MPC computation.

Stage 1: Token Dispersion. Because the input to an MPC protocol $\mathtt{Input} = X$ is divided equally among machines $1, \dots, \lceil MmH/s \rceil$, the first round of MPC computation routes each input token X_i to its respective token machine. This is completed by setting $(i, X_i) \in \mathtt{MachineOut}_{i'}^{(1)}$ if $(i, X_i) \in \mathtt{MachineIn}_{i'}^{(1)}$. Thus, $\mathtt{MachineIn}_i^{(2)} = \{(\mathtt{Src}, X_i)\}$ for all token machines $i \in [M]$.

Stage 2: Embedding Propagation. In rounds $2, \dots, D+1$, π computes the respective key, query, and value embeddings in each token machine and propagate them to respective inner product machines using the query and key/value propagation machines. Concretely:

- In round 2, each token machine i (whose memory contains X_i) computes m-dimensional embeddings embeddings $Q_i := (Q_{h,i}(X_i))_{h\in[H]}, K_i := (K_{h,i}(X_i))_{h\in[H]}, V_i := (V_{h,i}(X_i))_{h\in[H]}$. It transmits each embedding to the respective depth-1 query and key/value propagation machine nodes, while also preserving knowledge of its own X_i. (In all further rounds, we assume that $((i, X_i)) \in \mathtt{MachineOut}_i^{(r)}$ to ensure that token machine i can compute the skip-level connection at the end.) That is,

$$\begin{aligned}\mathtt{MachineOut}_i^{(2)} = \{&(i, X_i)\} \\ &\cup \{(\mathtt{ID}(\mathtt{Q}, i, 1, k'), Q_i) : k' \in \mathtt{Children}_1\} \\ &\cup \{(\mathtt{ID}(\mathtt{KV}, i, 1, k'), (K_i, V_i)) : k' \in \mathtt{Children}_1\}\,.\end{aligned}$$

Note that the total amount of messages sent is $b \cdot mH + 2b \cdot mH + m \le s$ and that the only machines receiving messages are size m-messages by token machines and size $\le 4mH$ messages by query and key/value propagation machines.

- In rounds $r \in \{3, \dots, D\}$, each query and key/value propagation machine of depth $d = r-2$ passes embeddings onto their successors. That is,

$$\begin{aligned}\mathtt{MachineOut}_{\mathtt{ID}(\mathtt{Q},i,d,k)}^{(r)} &= \{(\mathtt{ID}(\mathtt{Q}, i, d+1, k'), Q_i) : k' \in \mathtt{Children}_k\}\,, \\ \mathtt{MachineOut}_{\mathtt{ID}(\mathtt{KV},i,d,k)}^{(r)} &= \{(\mathtt{ID}(\mathtt{KV}, i, d+1, k'), (K_i, V_i)) : k' \in \mathtt{Children}_k\}\,.\end{aligned}$$

- In round $D+1$, the depth-$(D-1)$ query and key/value propagation machines pass their embeddings onto their respective inner product machines. That is,

$$\begin{aligned}\mathtt{MachineOut}_{\mathtt{ID}(\mathtt{Q},i,D-1,k)}^{(D+1)} &= \{(\mathtt{ID}(i, k'), Q_i) : k' \in \mathtt{Children}_k \cap [M]\}\,, \\ \mathtt{MachineOut}_{\mathtt{ID}(\mathtt{KV},i,D-1,k)}^{(D+1)} &= \{(\mathtt{ID}(k', i), (K_i, V_i)) : k' \in k' \in \mathtt{Children}_k \cap [M]\}\,.\end{aligned}$$

Stage 3: Softmax Computation. In rounds $D+2, \dots, 2D+2$, computes each inner product and iteratively builds up each attention output by accumulating partial softmax computations. For each query propagation machine $(\mathtt{Q}, i, d, k)$ and $h \in [H]$, we let $S_{i,d,k,h}$ and $Z_{i,d,k,h}$ denote its partial normalization and softmax computations respectively. That is,

$$\begin{aligned}
Z_{i,d,k,h} &= \sum_{i' \in \mathtt{Descendants}_{d,k}} \exp(Q_{h,i}(X_i)^\intercal K_{h,i'}(X_{i'})) \mathbb{1}\left\{\Lambda_{i,i'} = 0\right\} \\
&= \begin{cases} \sum_{k' \in \mathtt{Children}_k} Z_{i,d+1,k',h} & \text{if } d \le D-1, \\ \exp(Q_{h,i}(X_i)^\intercal K_{h,k}(X_k)) \mathbb{1}\left\{\Lambda_{i,k} = 0\right\} & \text{if } d = D. \end{cases} \\
S_{i,d,k,h} &= \frac{1}{Z_{i,d,k,h}} \sum_{i' \in \mathtt{Descendants}_{d,k}} \exp(Q_{h,i}(X_i)^\intercal K_{h,i'}(X_{i'})) V_{h,i'}(X_{i'}) \mathbb{1}\left\{\Lambda_{i,i'} = 0\right\} \\
&= \begin{cases} \sum_{k' \in \mathtt{Children}_k} \frac{Z_{i,d+1,k',h}}{Z_{i,d,k,h}} \cdot S_{i,d+1,k',h} & \text{if } d \le D-1, \\ V_{h,k}(X_k) \mathbb{1}\left\{\Lambda_{i,k} = 0\right\} & \text{if } d = D; \end{cases}
\end{aligned}$$

Note that $S_{i,0,1,h} = (\mathrm{softmax}(Q_h(X)K_h(X)^\intercal + \Lambda_h)V_h(X))_i$ and let $S_{i,d,k} = (S_{i,d,k,h})_{h \in [H]} \in \mathbb{R}^{H \times m}$ and $Z_{i,d,k} = (Z_{i,d,k,h})_{h \in [H]} \in \mathbb{R}^H$

- In round $D+2$, each inner product machine computes its respective inner products and passes its partial softmax computations to its parent query propagation machine. As a result of round $D+1$, each inner product machine (i, i') recently received the embeddings necessary to compute the relevant inner product:

  $$\mathtt{MachineIn}^{(d+2)}_{\mathtt{ID}(i,i')} = \left\{(\mathtt{ID}(\mathtt{Q}, i, D-1, \mathtt{Parent}_i), Q_i), (\mathtt{ID}(\mathtt{KV}, i', D-1, \mathtt{Parent}_{i'}), (K_{i'}, V_{i'}))\right\}.$$

  It propagates the respective partial computations $S_{i,D,i'}$ and $Z_{i,D,i'}$ as follows:

  $$\mathtt{MachineOut}^{(D+2)}_{\mathtt{ID}(i,i')} = \left\{(\mathtt{ID}(\mathtt{Q}, i, D-1, \mathtt{Parent}_i), (S_{i,D,i'}, Z_{i,D,i'}))\right\}.$$

  Note that each depth-$(D-1)$ query propagation machine receives messages of size at most $b \cdot (m+1)H \le s$.

- In rounds $r \in \{D+3, \dots, 2D\}$, partial softmax computations are received by query propagation machines of depth $d = 2D+1-r$, added together, and passed along to their parent machines. That is, given

  $$\mathtt{MachineIn}^{(r)}_{\mathtt{ID}(\mathtt{Q},i,d,k)} = \left\{(\mathtt{ID}(\mathtt{Q}, i, d+1, k'), (S_{i,d+1,k'}, Z_{i,d+1,k'})) : k' \in \mathtt{Children}_k\right\},$$

  each respective machine computes $S_{i,d,k}$ and $Z_{i,d,k}$ recursively and propagates

  $$\mathtt{MachineOut}^{(r)}_{\mathtt{ID}(\mathtt{Q},i,d,k)} = \left\{(\mathtt{ID}(\mathtt{Q}, i, d-1, \mathtt{Parent}_k), (S_{i,d,k}, Z_{i,d,k})\right\}.$$

- In round $2D+1$, the top-most query propagation tokens pass their partial sums to the token machines:

  $$\mathtt{MachineOut}^{(2D+1)}_{\mathtt{ID}(\mathtt{Q},i,1,k)} = \left\{(i, (S_{i,1,k}, Z_{i,1,k}))\right\}.$$

- In round $2D+2$, the token machines compute their respective output of the transformer, $T(X)_i$. Given input

  $$\mathtt{MachineIn}^{(2D+2)}_i = \left\{(k', (S_{i,1,k'}, Z_{i,1,k'})) : k' \in \mathtt{Children}_1\right\} \cup \left\{(i, X_i)\right\},$$

  the token machine i computes $S_{i,0,1}$ and $H_{i,0,1}$ and then

  $$T(X)_i = \psi_i\left(X_i + \sum_{h=1}^H \mathrm{softmax}(Q_h(X)K_h(X)^\intercal + \Lambda_h)_i^\intercal V_h(X)\right) = \psi_i\left(X_i + \sum_{h=1}^H S_{i,0,1,h}\right).$$

  This quantity is used as an intermediate product for the final phase of computation.

Stage 4: Token Compression. We invert Stage 1 by properly compressing the MPC output in the final round $2D+3$. That is, we let $\texttt{MachineOut}_i^{(2D+2)} = \{(\lfloor imH/s \rfloor + 1, T(X)_i)\}$ for each token machine $i \in [M]$, which ensures that the outputs are condensed in the proper order in machines $1, \dots, \lceil MmH/s \rceil$.

Precision Analysis. In order for the proof to be fully sound, care must be taken to ensure that the computation of each self-attention output $S_{i,0,1,h}$ is handled with proper numeric precision, as discussed in Appendix A. We show that each $S_{i,0,1,h}$ is a *valid implementation* of its corresponding self-attention unit, per Definition A.1.

To do so, we let $\hat{S}_{i,d,k,h}$ and $\hat{Z}_{i,d,k,h}$ denote the p-bit representations of $S_{i,d,k,h}$ and $Z_{i,d,k,h}$, where scalars of $\hat{S}_{i,d,k,h}$ and $\log(\hat{Z}_{i,d,k,h})$ are represented as discretized rational numbers z satisfying $|z| \leq \frac{1}{2}2^{p/2}$ and $z \cdot 2^{p/2} \in \mathbb{Z}$. For some sufficiently small $p' = \Theta(p)$, we assume that all embeddings $Q_h(X), K_h(X), V_h(X)$ have scalars z satisfying $|z| \leq \frac{1}{2}2^{p'/2}$ and $z \cdot 2^{p'/2} \in \mathbb{Z}$. We prove that for each $h \in [H]$,

$$\left\| S_{i,0,1,h} - \hat{S}_{i,d,k,h} \right\|_\infty = O\left(\frac{1}{2^{p'}}\right).$$

Boundedness of intermediate representations is not an issue because

$$\log(Z_{i,d,k,h}) \leq O(\log(N) + \max_{i,i'} |Q(X)_i^\mathsf{T} K(X)_{i'}|) = \exp(O(p')),$$

and

$$\|S_{i,d,k,h}\|_\infty \leq \|V(X)\|_\infty \leq 2^{p'/2}.$$

It remains to show that that all intermediate representations are sufficiently close to their exact counterparts. We prove the following via an inductive argument for $d = D, D-1, \dots, 0$:

$$\left| \log(Z_{i,d,k,h}) - \log(\hat{Z}_{i,d,k,h}) \right| \leq \frac{(2b)^{D-d}}{2^{p/2}}, \tag{2}$$

$$\left\| S_{i,d,k,h} - \hat{S}_{i,d,k,h} \right\|_\infty \leq \frac{2^{p'/2}(8b)^{D-d}}{2^{p/2}}. \tag{3}$$

If (3) holds for $d = 0$, then the claim holds for sufficiently large $p = \Theta(p')$.

For the base case D, we verify (3) by

$$\left\| S_{i,D,k,h} - \hat{S}_{i,D,k,h} \right\|_\infty = \left\| V_{h,k}(X_k) \mathbb{1}\{\Lambda_{i,k} = 0\} - \hat{S}_{i,D,k,h} \right\|_\infty \leq \frac{1}{2^{p/2}},$$

due to the ability to access $V_{h,k}(X_k)$ and round it directly. We verify (2) due to the immediate access to and boundedness of $Q_{h,i}(X_i)^\mathsf{T} K_{h,k}(X_k)$:

$$|\log(Z_{i,d,k,h})| \leq \left| Q_{h,i}(X_i)^\mathsf{T} K_{h,k}(X_k) \right| \leq \|Q_{h,i}(X_i)\|_2 \, \|K_{h,k}(X_k)\|_2 \leq N \cdot 2^{p'/2}.$$

We prove the inductive step for $d-1$, assuming that the inductive hypothesis holds for d. We first address $\hat{Z}_{i,d-1,k,h}$ by employing the Lipschitzness of the log-sum-exp function.

$$\begin{aligned}
\left| \log(Z_{i,d-1,k,h}) - \log(\hat{Z}_{i,d-1,k,h}) \right| &\leq \frac{1}{2^{p/2}} + \left| \log\left(\sum_{k'} \exp(\log(Z_{i,d,k',h})) \right) - \log\left(\sum_{k'} \exp(\log(\hat{Z}_{i,d,k',h})) \right) \right| \\
&\leq \frac{1}{2^{p/2}} + \sum_{k'} \left| \log(Z_{i,d,k',h}) - \log(\hat{Z}_{i,d,k',h}) \right| \\
&\leq \frac{1}{2^{p/2}} + b \cdot \frac{(2b)^{D-d}}{2^{p/2}} \leq \frac{(2b)^{D-d+1}}{2^{p/2}}.
\end{aligned}$$

To obtain (3) for $d-1$, we first note that for sufficiently large p:

$$\begin{aligned}\left|1-\frac{\hat{Z}_{i,d,k',h}Z_{i,d-1,k',h}}{Z_{i,d,k,h}\hat{Z}_{i,d-1,k',h}}\right| &= \left|1-\exp\left(\log\left(\frac{\hat{Z}_{i,d,k',h}}{Z_{i,d,k',h}}\right)+\log\left(\frac{Z_{i,d-1,k,h}}{\hat{Z}_{i,d-1,k,h}}\right)\right)\right| \\ &\leq 1+2\left(\left|\log\frac{\hat{Z}_{i,d,k',h}}{Z_{i,d,k',h}}\right|+\left|\log\frac{Z_{i,d-1,k,h}}{\hat{Z}_{i,d-1,k,h}}\right|\right) \\ &\leq \frac{4\cdot(2b)^{D-d+1}}{2^{p/2}}.\end{aligned}$$

We conclude by using the fact that each $S_{i,d-1,k,h}$ is a convex combination of other $S_{i,d,k,h}$.

$$\begin{aligned}\left\|S_{i,d-1,k,h}-\hat{S}_{i,d-1,k,h}\right\|_\infty &\leq \frac{1}{2^{p/2}}+\sum_{k'}\left\|\frac{Z_{i,d,k',h}}{Z_{i,d-1,k',h}}S_{i,d,k',h}-\frac{\hat{Z}_{i,d,k',h}}{\hat{Z}_{i,d-1,k',h}}\hat{S}_{i,d,k',h}\right\|_\infty \\ &\leq \frac{1}{2^{p/2}}+\sum_{k'}\frac{Z_{i,d,k',h}}{Z_{i,d-1,k',h}}\left\|S_{i,d,k',h}-\frac{\hat{Z}_{i,d,k',h}Z_{i,d-1,k',h}}{Z_{i,d,k,h}\hat{Z}_{i,d-1,k',h}}\hat{S}_{i,d,k',h}\right\|_\infty \\ &\leq \frac{1}{2^{p/2}}+\sum_{k'}\frac{Z_{i,d,k',h}}{Z_{i,d-1,k',h}}\left\|S_{i,d,k',h}-\hat{S}_{i,d,k',h}\right\|_\infty \\ &\quad +\sum_{k'}\frac{Z_{i,d,k',h}}{Z_{i,d-1,k',h}}\left\|\hat{S}_{i,d,k',h}\right\|_\infty\left|1-\frac{\hat{Z}_{i,d,k',h}Z_{i,d-1,k',h}}{Z_{i,d,k,h}\hat{Z}_{i,d-1,k',h}}\right| \\ &\leq \frac{1}{2^{p/2}}+\frac{2^{p'/2}(8b)^{D-d}}{2^{p/2}}+2^{p'/2}\sum_{k'}\frac{Z_{i,d,k',h}}{Z_{i,d-1,k',h}}\left|1-\frac{\hat{Z}_{i,d,k',h}Z_{i,d-1,k',h}}{Z_{i,d,k,h}\hat{Z}_{i,d-1,k',h}}\right| \\ &\leq 2\cdot\frac{2^{p'/2}(8b)^{D-d}}{2^{p/2}}+2^{p'/2}\cdot\frac{4\cdot(2b)^{D-d+1}}{2^{p/2}}\leq\frac{2^{p'/2}(8b)^{D-d+1}}{2^{p/2}}.\end{aligned}$$

Owing to the fact that D and p' are constants and $b = N^{O(1)}$, a sufficiently large choice of p guarantees that the implementation is valid. □

C.2 Proof of Corollary 3.5

Corollary 3.5. *Let $\epsilon \in (0,1)$ be any constant, and let $D \geq N^\epsilon$. Assume Conjecture 2.4, and suppose there exists $T \in \mathsf{Transformer}^N_{m,L,H}$ with $mH = O(D^{1-\epsilon})$ that decides connectivity of any input graph with connected components having diameter $\leq D$. Then $L = \Omega(\log D)$.*

We prove Corollary 3.5 by combining Theorem C.1 and Conjecture 2.4.

Proof. Fix any $D \leq N$ with $D \geq N^\xi$ for some $\xi \in (0,1]$. Let C_1 denote a cycle graph on D vertices, and let C_2 denote the union of two cycle graphs each with $D/2$ vertices.

Suppose there is a transformer $T \in \mathsf{Transformer}^N_{m,L,H}$ with $mH = O(D^{1-\epsilon})$ that determines the connectivity of graphs with at most N edges and connected components with diameter at most D. We will show that it can be used to design an $\Theta(L)$-round MPC protocol π that distinguishes graphs C_1 and C_2 with $n = D$ edges.

Let π' be an MPC protocol that exactly computes the output of T using taking $R = O(L)$ rounds with local memory $s = O(D^{1-\epsilon/2})$ and $q = O(N^2)$ machines, which is guaranteed to exist by Theorem C.1.

Let $n := 2\left\lfloor\frac{D}{4}\right\rfloor$ and $k := \left\lfloor\frac{N}{n}\right\rfloor$. We design π with the same local memory and machine count to determine the identity of input graph $G = (V,E) \in \{C_1, C_2\}$ provided as an arbitrary sequence of n edges. Let $u \in V$ be an arbitrary vertex in G.

Using a constant number of MPC rounds, π converts G into a graph $G' = (V', E')$ with $|E'| = kn + k \leq N$ and diameter $n + 2 \leq D$ such that G' is connected if and only if $G = C_1$. We do so by letting G' be composed of k copies $G^1, \dots, G^k$ of G on separate vertices, along with k extra edges connecting the vertex corresponding to u in each G^j (say $u^j \in G^j$) to

$u^1 \in G_1$. This ensures that the connectivity correspondence and edge count diameter bounds are met. Since G' can be produced by simply copying edges from G and adding an additional edge each time an edge containing u is copied, π can produce G' in $O(1)$ rounds.

Then, π simulates π' on G' and returns its output. Since G' is connected if and only if $G = C_1$, this protocol suffices to distinguish C_1 and C_2. Because the protocol uses $s = O(n^{1-\epsilon/2})$ local memory and $q = O(n^{2/\xi})$ machines, Conjecture 2.4 implies that π (and hence T) only exists if $L = \Omega(\log n) = \Omega(\log N)$. □

D Proofs from Section 4.1

D.1 Proof of Theorem 4.2

Theorem 4.2. *For any $k \in \mathbb{N}$ and alphabet Σ with $|\Sigma| \leq N$, there exists $T \in \mathsf{MaskTransformer}^N_{m,L,H}$ that computes $\mathrm{hop}_k \colon \Sigma^N \to (\Sigma \cup \{\perp\})^N$ with $m = O(1)$, $L = \lfloor \log_2 k \rfloor + 2$, and $H = 1$.*

Proof. We design a masked transformer that implements hop_k in two phases. The first two layers compute $\mathrm{find}^1_X(i)$ for each $i \in [N]$ using a similar approach to the induction heads construction of (Bietti et al., 2023). The subsequent layers employ a doubling trick to compute each $\mathrm{find}^{2^{\ell-2}}_X(i)$ after ℓ layers.

To do so we employ two technical lemmas (which are proved in Appendix F.4) that describe the implementation of masked self-attention units that copy .

Lemma D.1. *For some $m \geq d+2$, $\tau : [N] \times \mathbb{R}^m \to [N]$, and $\rho : \mathbb{R}^m \to \mathbb{R}^d$, there exists an attention head $\mathrm{lookUp}_{\tau,\rho} \in \mathsf{MaskAttn}^N_m$ with precision $p = O(\log N)$ and $m \geq d+2$ satisfying $\mathrm{lookUp}_{\tau,\rho}(X)_{i,:d} = \rho(X_{\tau(i,X_i)})$.*

Lemma D.2. *For any finite alphabet Σ, $m \geq d+2$, $\mu_1, \mu_2 : \mathbb{R}^m \to \Sigma$, and $\rho : \mathbb{R}^m \to \mathbb{R}^d$, there exists an attention head $\mathrm{lastOccurrence}_{\mu,\rho} \in \mathsf{MaskAttn}^N_m$ with precision $p = O(\log(N\,|\Sigma|))$ such that,*

$$\mathrm{lastOccurrence}(X)_{i,:d} = \begin{cases} \rho(\vec{0}) & \textit{if}\, \forall\, i' < i : \mu_1(X_{i'}) \neq \mu_2(X_i), \\ \rho(X_{i'}) & \textit{if}\, i' = \max\{i' < i : \mu_1(X_{i'}) = \mu_2(X_i)\}\,. \end{cases}$$

The first layer obtains the previous token X_{i-1} from each X_i. This is accomplished via the self-attention head $\mathrm{lookUp}_{\tau,\rho}$ with $\tau(i, X_i) = i-1$ and $\rho(X_i) = X_i$.

The second layer retrieves $(\mathrm{find}^1_X(i), X_{\mathrm{find}^1_X(i)})$ for each $i \in [N]$ by finding the most recent token whose *preceding* token is X_i. It does so by employing the $\mathrm{lastOccurrence}_{\mu_1,\mu_2,\rho}$ primitive on the intermediate state $X^1_i = (X_i, X_{i-1})$ with $\mu_1(X^1_i) = X_{i-1}$, $\mu_2(X^1_i) = X_i$, and $\rho(X^1_i) = (i, X_i)$.

- If $\mathrm{find}^1_X(i) > 0$, then $\mathrm{lastOccurrence}_{\mu_1,\mu_2,\rho}(X^1_i) = (\mathrm{find}^1_X(i), X_{\mathrm{find}^1_X(i)})$.
- Otherwise, it obtains $\vec{0}$ and performs no further passing, returning $\perp$ after all L layers.

If $k = 1$, the transformer returns $T(X)_i = X_{\mathrm{find}^1_X(i)} = \mathrm{hop}_k(X)_i$.

Otherwise, let $k := \sum_{j=0}^{\lfloor \log_2 k \rfloor} k_j 2^j$ for some $k_j \in \{0, 1\}$, and let $k_{:\ell} = \sum_{j=0}^{\ell} k_j 2^j$. Construct a transformer inductively to ensure that the ith output of the ℓth layer $X^\ell_i \in \mathbb{R}^m$ for $\ell \geq 2$ contains an encoding of

$$\left(X_i, \mathrm{find}^{2^{\ell-2}}_X(i), X_{\mathrm{find}^{2^{\ell-2}}_X(i)}, \mathrm{find}^{k_{:\ell-2}}_X(i), X_{\mathrm{find}^{k_{:\ell-2}}_X(i)}\right).$$

Note that the base case holds for $\ell = 2$, since $\mathrm{find}^{k_{:0}}_X(0) = \mathrm{find}^1_X(0)$ if $k_0 = 0$ and is i otherwise.

For each $\ell = 1, \ldots, \lfloor \log_2 k \rfloor + 1$, we assume that the inductive hypothesis holds up to layer ℓ and prove that it also holds for layer $\ell + 1$. To do so, we use a $\mathrm{lookUp}_{\tau,\rho}$ self-attention head with $\tau(i, X^\ell_i) = \mathrm{find}^{2^{\ell-2}}_X(i)$ and

$$\rho(X^\ell_i) = (\mathrm{find}^{2^{\ell-2}}_X(i), X_{\mathrm{find}^{2^{\ell-2}}_X(i)}, \mathrm{find}^{k_{:\ell-2}}_X(i), X_{\mathrm{find}^{k_{:\ell-2}}_X(i)}),$$

which ensures that $X_i^{\ell+1}$ can encode

$$\begin{aligned}
\mathrm{find}_X^{2^{\ell-1}}(i) &= \mathrm{find}_X^{2^{\ell-2}}(\mathrm{find}_X^{2^{\ell-2}}(i)) \\
X_{\mathrm{find}_X^{2^{\ell-1}}(i)} &= X_{\mathrm{find}_X^{2^{\ell-2}}(\mathrm{find}_X^{2^{\ell-2}}(i))} \\
\mathrm{find}_X^{k_{:\ell-1}}(i) &= \begin{cases} \mathrm{find}_X^{k_{:\ell-2}}(\mathrm{find}_X^{2^{\ell-2}}(i)) & \text{if } k_{\ell-1} = 1 \\ \mathrm{find}_X^{k_{:\ell-2}}(i) & \text{if } k_{\ell-1} = 0 \end{cases} \\
X_{\mathrm{find}_X^{k_{:\ell-1}}(i)} &= \begin{cases} X_{\mathrm{find}_X^{k_{:\ell-2}}(\mathrm{find}_X^{2^{\ell-2}}(i))} & \text{if } k_{\ell-1} = 1 \\ X_{\mathrm{find}_X^{k_{:\ell-2}}(i)} & \text{if } k_{\ell-1} = 0. \end{cases}
\end{aligned}$$

As a result, the output of layer $L = \lfloor \log_2 k \rfloor + 2$ contains an encoding of

$$X_{\mathrm{find}_X^{k_{:L-2}}(i)} = X_{\mathrm{find}_X^{k}(i)} = \mathrm{hop}_k(X)_i$$

for each $i \in [N]$. This is returned as the output of $T(X)$.

□

D.2 Proof of Corollary 4.3

Corollary 4.3. *Assuming Conjecture 2.4, for any constants $\xi \in (0, 1/2]$ and $\epsilon \in (0, 1)$, and any even $k = \Theta(N^\xi)$, every transformer $T \in \mathsf{MaskTransformer}_{m,L,H}^N$ with $mH = O(k^{1-\epsilon})$ that computes hop_k has depth $L = \Omega(\log k)$.*

Proof. The proof is analogous to that of Corollary 3.5. Let C_1 be a cycle on k vertices, and C_2 be the union of two cycles each on $k/2$ vertices. So both C_1 and C_2 have k edges. We show that the existence of $T \in \mathsf{Transformer}_{m,L,H}^N$ with $mH = O(k^{1-\epsilon})$ such that $T(X) = \mathrm{hop}_k(X)$ can be used to design an $\Theta(L)$-round MPC protocol π to solve the task.

As a result of Theorem C.1, there exists an MPC protocol π' that exactly computes T with $R = \Theta(L)$ rounds with local memory $s = O(D^{1-\epsilon/2})$ and $q = O(N^2)$ machines. On input $G = (V, E) \in \{C_1, C_2\}$, we design a constant-round protocol that computes an sequence $X \in \Sigma^N$ such that $\mathrm{hop}_k(X)_N$ exactly determines the identity of G.

Since the k edges are passed to π in an unknown ordering with unknown labelings, we let $V = [k]$ and denote the edges as $e_1 = \{u_1, v_1\}, \dots, e_k = \{u_k, v_k\}$. We define an operator next over the domain $\{(u, v), (v, u) : \{u, v\} \in E\}$ as follows: for $\{u, v\} \in E$, let $\mathrm{next}(u, v) := (v', u)$ where $v' \in V$ is the unique vertex $v' \neq v$ such that $\{u, v'\} \in E$. Notice that next is well-defined because all vertices in a cycle have degree 2. If $G = C_2$, then $\mathrm{next}^{k/2}(u_i, v_i) = (u_i, v_i)$ for any $i \in [k]$.

To set up our encoding of G as a sequence X, we first construct a gadget for each edge e_i that will be used to compute a single $\mathrm{next}(u_i, v_i)$. Under the alphabet $\Sigma = [k] \cup \{\dagger, \star, \_\}$, we define the nine-token sequence

$$\mathbf{e}_i = \ \star\, u_i \ \dagger \ v_i \ u_i \ \dagger \ v_i \ \star \ \_.$$

This gadget ensures that two hops will swap the values of u_i and v_i. That is

$$\begin{aligned}
&\mathrm{find}^2_{\mathbf{e}_i \circ u_i}(10) = \mathrm{find}^1_{\mathbf{e}_i \circ u_i}(6) = 4, && X_{\mathrm{find}^2_{\mathbf{e}_i \circ u_i}(10)} = v_i, \\
&\mathrm{find}^2_{\mathbf{e}_i \circ v_i}(10) = \mathrm{find}^1_{\mathbf{e}_i \circ v_i}(8) = 2, && X_{\mathrm{find}^2_{\mathbf{e}_i \circ v_i}(10)} = u_i.
\end{aligned}$$

Likewise, concatenating sequences corresponding to overlapping edges facilitates multiple hops. For example, if $e_1 = (1, 2), e_2 = (3, 4), e_3 = (2, 3)$, then

$$\begin{aligned}
&\mathrm{find}^2_{\mathbf{e}_1 \circ \mathbf{e}_2 \circ \mathbf{e}_3 \circ 2}(28) = 22, && X_{\mathrm{find}^2_{\mathbf{e}_1 \circ \mathbf{e}_2 \circ \mathbf{e}_3 \circ 2}(28)} = 3, \\
&\mathrm{find}^4_{\mathbf{e}_1 \circ \mathbf{e}_2 \circ \mathbf{e}_3 \circ 2}(28) = 13, && X_{\mathrm{find}^4_{\mathbf{e}_1 \circ \mathbf{e}_2 \circ \mathbf{e}_3 \circ 2}(28)} = 4, \\
&\mathrm{find}^4_{\mathbf{e}_1 \circ \mathbf{e}_2 \circ \mathbf{e}_3 \circ 3}(28) = 2, && X_{\mathrm{find}^4_{\mathbf{e}_1 \circ \mathbf{e}_2 \circ \mathbf{e}_3 \circ 3}(28)} = 1.
\end{aligned}$$

Let

$$\mathbf{E} := (\mathbf{e}_1 \circ \mathbf{e}_2 \circ \cdots \circ \mathbf{e}_k)^{k/2} \circ 1$$

be a length $N_k := 9k \cdot \frac{k}{2} + 1$ sequence and let $X = (\_)^{N-N_k} \circ \mathbf{E}$. We show that $\mathrm{hop}_k(X)_N = \mathrm{hop}_k(\mathbf{E})_{N_k} = 1$ if and only if $G = C_2$.

Without loss of generality, let $\{j, j+1\} = e_{i_j} \in E$ for all $j \in [\frac{k}{2} - 1]$. Let $e_{i_0} = \{1, v^*\}$, where $v^* = \frac{k}{2}$ if $G = C_2$ and $v^* = k$ if $G = C_1$. Assume without loss of generality that $i_1 > i_0$. We argue inductively that for any $j \in [\frac{k}{2}]$:

1. Every two hops simulates a single step of next:
$$\mathrm{hop}_{2j}(\mathbf{E})_{N_k} = \mathrm{next}^j(1, v^*)_1 = \begin{cases} j & \text{if } j+1 < \frac{k}{2} \text{ or } G = C_1, \\ 1 & \text{if } j = \frac{k}{2},\ G = C_2; \end{cases}$$

2. Every two hops never "jumps" by more than one repetition of all edges gadgets:
$$\mathrm{find}_{\mathbf{E}}^{2j}(N_k) \geq \mathrm{find}_{\mathbf{E}}^{2j-2}(N_k) - 9(k-1);$$

3. The executed gadget corresponds to the correct edge and the gadget is executed correctly:
$$\mathrm{find}_{\mathbf{E}}^{2j}(N_k) \in \{9kj' + 9i_j + \iota : j' \in \mathbb{N}, \iota \in \{2, 4\}\}.$$

If all three conditions are met, then $\mathrm{hop}_k(X)_N = 1$ if and only if $G = C_1$ from condition 1.

We first show that the base case holds for $j = 1$. Since $i_1 > i_0$, the second-last time 1 appears in the $\mathbf{E}$ is in the final encoding $\mathbf{e}_{i_1}$. By the two-case analysis of the $\mathbf{e}_{i_1}$ gadget, we validate that $\mathrm{hop}_2(\mathbf{E})_{N_k} = 2$ and conditions (1) and (3) hold. Since $\mathbf{e}_{i_1}$ cannot be the first edge encoding appearing in $\mathbf{e}_1 \circ \mathbf{e}_2 \circ \cdots \circ \mathbf{e}_k$, owing to it following $\mathbf{e}_{i_0}$), condition (2) is satisfied.

Suppose that the inductive hypotheses holds up to $j < \frac{k}{2}$. Then, we argue that it holds for $j+1$. Since $\mathrm{hop}_{2j}(\mathbf{E})_{N_k} = j+1$ (from condition (1)) and $\mathrm{find}_{\mathbf{E}}^{2j}(N_k)$ resides at the left-most side of the gadget for $\mathbf{e}_{i_j}$ (from condition (3)), the two subsequent $\mathrm{find}_{\mathbf{E}}$ iterations must occur in the gadget $\mathbf{e}_{i_{j+1}}$. Because $\mathrm{find}_{\mathbf{E}}^{2j}(N_k) \geq 9k(k-j)$ (from condition (2)), all edges appear in the k gadgets to the left of $\mathrm{find}_{\mathbf{E}}^{2j}(N_k)$, and all other edges (including $\mathbf{e}_{i_{j+1}}$) must occur before the next occurrence of $\mathbf{e}_{i_j}$. Thus, the two hops occur in the $\mathbf{e}_{i_{j+1}}$ gadget (within distance $9(k-1)$) and results in a properly positioned $\mathrm{find}_{\mathbf{E}}^{2j+2}(N_k)$ with $\mathrm{hop}_{2j+2}(\mathbf{E})_{N_k} = \mathrm{next}^{j+1}(1, v^*)_1$.

Since an MPC protocol can convert G to X using a constant number of layers, and because π' outputs $T(X)_N = 1$ if and only if $G = C_1$, we can construct a protocol of π by simulating π'. Because the protocol π uses $s = O(k^{1-\epsilon/2})$ local memory and $q = O(k^{2/\xi})$ machines, Conjecture 2.4 implies that the existence of T requires $L = \Omega(\log k)$. □

E Proofs from Section 5

E.1 Multi-Player Pointer Chasing Communication Complexity

We introduce the multi-pass multi-player blackboard communication model studied by Guha & McGregor (2009) and Assadi & N (2021) to prove lower bounds for multi-pass streaming algorithms. A protocol in this model specifies how k players, each possessing a portion of a shared input, can jointly compute a function on the input over the course of R rounds of communication. In each round, all players take turns to broadcast an s-bit message to all other players. We provide a formal definition of the model as described in Section 6 of Assadi & N (2021).

Definition E.1. A *k-player R-round s-space sequential blackboard communication protocol* includes k players $P_1, \ldots, P_k$. On input Z that can be partitioned into $(Z_1, \ldots, Z_k)$, each player P_j is provided with its respective Z_j. In each round, players communicate via a shared blackboard. That is, in round r and in order $P_k, \ldots, P_1$, each player P_j writes a message $\Pi_j^r \in \{0,1\}^s$ on the blackboard (which can be viewed by all players) as a potentially randomized function of input Z_j and all information on the blackboard. After the conclusion of R rounds, the final message Π_1^R is the output of the protocol.

Assadi & N (2021) proves a lower bound on the round complexity necessary to solve the well-studied *multi-party pointer chasing problem* of Nisan & Wigderson (1993). We present the problem as defined by Assadi & N (2021).

Definition E.2. For $q, k \in \mathbb{Z}_+$, let an (q, k)-*layered graph* $G = (V, E)$ have disjoint vertex layers $V_1, \dots, V_{k+1}$ with $V = V_1 \cup \dots \cup V_{k+1}$ and each $|V_j| = q$ and edge layers $E_1, \dots, E_k$ with $E = E_1 \cup \dots \cup E_k$ and each E_j being a perfect matching between V_j and V_{j+1}. The *pointer chasing* task is provides a (q, k)-layered graph G, an arbitrary $v \in V_1$, and an arbitrary equipartition V_{k+1}^1 and V_{k+1}^2 of V_{k+1} as input and asks whether v is connected to a vertex in V_{k+1}^1 or V_{k+1}^2.

Assadi & N (2021) give the following lower bound.

Proposition E.3 (Proposition 4.12 of Assadi & N, 2021)**.** *Consider a k-player R-round s-space sequential blackboard protocol that solves the (q, k)-pointer chasing task where each player P_j is provided with the matching E_j and v and V_{k+1}^1, V_{k+1}^2 are globally known. Then, the protocol succeeds with probability at least $\frac{2}{3}$ only if $R \geq k$ or $s = \Omega(\frac{q}{k^5})$.*

All of the lower bounds in Section 5 are most naturally proved by reducing from hop_k, rather than pointer chasing. So we first prove a lower bound for hop_k using the lower bound for pointer chasing from Proposition E.3.

Proposition E.4. *Consider a k-player R-round s-space sequential blackboard protocol that computes $\mathrm{hop}_k(X)_N$ on any $X \in \Sigma^N$ for $\Sigma = [2q + 2]$ with $q = \lfloor \frac{N}{2k} \rfloor$ where each player P_j is provided with $X^j := (X_{2(k-j)q+1}, \dots, X_{2(k-j+1)q})$, except for P_1, who is given $X^1 := (X_{2(k-1)q+1}, \dots, X_N)$. Then, the protocol succeeds with probability at least $\frac{2}{3}$ only if $R \geq k$ or $s = \Omega(\frac{N}{k^6})$.*

Proof. Assuming the existence of a k-player R-round s-space sequential blackboard protocol for $\mathrm{hop}_k(X)_N$ as described above, we design a protocol for solving (q, k)-pointer chasing with R rounds and s-size messages. The claimed lower bound will then follow by Proposition E.3.

Consider any pointer chasing input with universally known $V_1, \dots, V_{k+1}$, $v \in V_1$, and V_{k+1}^1 and V_{k+1}^2, and each player P_j knowing matching E_j. We recursively define $v_1, \dots, v_{k+1}$ such that $v_1 = v$ and $(v_j, v_{j+1}) \in E_j$, noting that the output hinges on whether $v_{k+1} \in V_{k+1}^1$.

Without loss of generality, let $v = 1$ and

$$V_j = \begin{cases} \{1, \dots, q\} & \text{if } j \text{ is odd,} \\ \{q + 1, \dots, 2q\} & \text{if } j \text{ is even.} \end{cases}$$

Each player independently determines their substring X^j of a input X to hop_k before running the aforementioned protocol:

- Player P_1 encodes X^1 by letting $X_N = s = 1$ and for any $i \in 1, \dots, 2q$, letting

  $$X_i^1 = \begin{cases} \frac{i+1}{2} \in V_1 & \text{if } i \text{ is odd,} \\ i' \in V_2 & \text{if } i \text{ is even, } (\frac{i}{2}, i') \in E_1. \end{cases}$$

  This ensures that that every integer in $\{1, \dots, 2q\}$ appears exactly once in $X_1^1, \dots, X_{2q}^1$, which in turn guarantees that $\mathrm{find}_X^1(N) = (k - 1 + 1)q + 2$ and that $X_{\mathrm{find}_X^1(N)} = v_2$ where $(1, i') \in E_1$.

- For any $j \in \{2, \dots, k - 1\}$, player P_j encodes E_j as X^j as follows. If j is odd, then for every $i \in \{1, \dots, 2q\}$,

  $$X_i^j = \begin{cases} \frac{i+1}{2} \in V_j & \text{if } i \text{ is odd,} \\ i' \in V_{j+1} & \text{if } i \text{ is even, } (\frac{i}{2}, i') \in E_j. \end{cases}$$

  Alternatively, if j is even,

  $$X_i^j = \begin{cases} q + \frac{i+1}{2} \in V_j & \text{if } i \text{ is odd,} \\ i' \in V_{j+1} & \text{if } i \text{ is even, } (q + \frac{i}{2}, i') \in E_j. \end{cases}$$

  Since every odd token corresponds to a vertex in V_j and each subsequent token corresponds to the vertex it's connected to by E_j, we can ensure that for every $i \in [2q]$:

  $$(X_{2(k-j+1)+i}, X_{\mathrm{find}_X^1(2(k-j+1)+i)}) \in E_j.$$

  Hence, it follows inductively that $X_{\mathrm{find}_X^j(N)} = v_{j+1}$.

- Player P_k encodes X^k if k is odd by letting

$$X_i^k = X_i = \begin{cases} \frac{i+1}{2} \in V_k & \text{if } i \text{ is odd,} \\ 2q+1 & \text{if } i \text{ is even, } (\frac{i}{2}, v) \in E_k, \text{ and } v \in V_{k+1}^1, \\ 2q+2 & \text{if } i \text{ is even, } (\frac{i}{2}, v) \in E_k, \text{ and } v \in V_{k+1}^2. \end{cases}$$

  Likewise, if k is even,

$$X_i^k = X_i = \begin{cases} q + \frac{i+1}{2} \in V_k & \text{if } i \text{ is odd,} \\ 2q+1 & \text{if } i \text{ is even, } (\frac{i}{2}, v) \in E_k, \text{ and } v \in V_{k+1}^1, \\ 2q+2 & \text{if } i \text{ is even, } (\frac{i}{2}, v) \in E_k, \text{ and } v \in V_{k+1}^2. \end{cases}$$

These jointly ensure that

$$\mathrm{hop}_k(X)_N = X_{\mathrm{find}_X^k(N)} = \begin{cases} 2q+1 & \text{if } v_{k+1} \in V_{k+1}^1, \\ 2q+2 & \text{if } v_{k+1} \in V_{k+1}^2. \end{cases}$$

Therefore, by formatting $E_1, \dots, E_k$ appropriately as X, running the protocol for $\mathrm{hop}_k(X)_N$, and observing that the final output of player P^1 is $2q+1$ if and only if $v_{k+1} \in V_{k+1}^1$, there exists a k-player R-round s-space protocol for pointer chasing. Hence, by Proposition E.3, the protocol for $\mathrm{hop}_k(X)_N$ must use $R \geq k$ rounds or $s = \Omega(\frac{N}{k^6})$ space. □

E.2 Proofs of Section 5.2

Corollary 5.2. *A multi-layer RNN of depth L and width m as above with $Y_N = \mathrm{hop}_k(X)_N$ satisfies either $L \geq k$ or $m = \Omega(\frac{N}{k^6})$.*

Proof. Suppose there exists a multi-layer RNN computing output Y with $Y_{N,1} = \mathrm{hop}_k(X)_N$ from input X with intermediate states $Z_1, \dots, Z_{L-1}$ and hidden states $H^1, \dots, H^L$. For any $\ell \in [L]$ and $i \leq i'$, note that $Z_i^\ell, \dots, Z_{i'}^\ell$ can be determined exactly from H_{i-1}^ℓ and $Z_i^{\ell-1}, \dots, Z_{i'}^{\ell-1}$. Given this RNN, we provide a multi-player blackboard communication protocol for solving $\mathrm{hop}_k(X)_N$ under the input model of Proposition E.4.

In round r, we assume inductively that each player P_j knows $Z^{\ell-1,j} = (Z^{\ell-1}_{2(k-j)q+1}, \dots, Z^{\ell-1}_{2(k-j+1)q})$, except for P_1, who knows $Z^{\ell-1,1} = (Z^{\ell-1}_{2(k-1)q+1}, \dots, Z_N^{\ell-1})$. In descending order, each player P_j computes $Z^{\ell,j}$ and $H^\ell_{2(k-j+1)q}$—writing the latter on the blackboard—from $Z^{\ell-1,j}$ and $H^\ell_{2(k-j)q}$,which was written on the blackboard by the previous player. Thus, player P^1 after round L knows and outputs $Z_{N,1}^L = Y_{N,1} = \mathrm{hop}_k(X)_N$, which provides an L-round protocol m-space protocol.

So the claimed lower bounds on width and depth follow from Proposition E.4. □

E.3 Proofs of Section 5.3

Corollary 5.3. *Any $T \in \mathsf{KernelFormer}_{m,m',L,H}^N$ with $T(X)_N = \mathrm{hop}_k(X)_N$ satisfies either $L \geq k$ or $mm'Hp = \Omega(\frac{N}{k^6})$.*

Proof. Under the distribution of input $X = (X^1, \dots, X^k)$ to players $P_1, \dots, P_k$ stipulated in the statement of Proposition E.4, we explain how the players can all compute the outcome of a single layer of H-headed kernelized attention in a single round of a blackboard protocol. It is immediate that a depth L network can be simulated in L rounds.

On input X, consider H kernelized self-attention units with embeddings $(Q_1', K_1', V_1), \dots, (Q_H', K_H', V_H)$ and output MLP ψ. Each player P_j immediately computes its embeddings $(Q_h'(X^j), K_h'(X^j), V_h(X^j))_{h \in [H]}$, followed by $(K_h'(X^j)^\mathsf{T} V_h(X^j)) \in \mathbb{R}^{m' \times m}$ for each $h \in [H]$. Because the object is to compute for each h

$$\psi(Q_h'(X) K_h'(X)^\mathsf{T} V_h(X)) = \psi(Q_h'(X) \sum_{j=1}^k K_h'(X^j)^\mathsf{T} V_h(X^j)),$$

each player writes their $(K'_h(X^j)^\mathsf{T} V_h(X^j))_{h \in [H]}$ using message size $s = \Theta(mm'Hp)$. Each can then construct $K'_h(X)^\mathsf{T} V_h(X))$ by reading the board, and use it to compute its respective outputs without requiring supplemental communication.

Hence, T (and thus $\mathrm{hop}_k(X)_N$) can be simulated using an L-round blackboard protocol with message size $s = \Theta(mm'Hp)$, and the corollary follows from Proposition E.4. □

Corollary 5.4. *Any $T \in \Lambda^{w,g}$-$\mathsf{Attn}^N_{m,L,H}$ with $T(X)_N = \mathrm{hop}_k(X)_N$ satisfies either $L \geq k$ or $(w + \frac{N}{gk})mHp = \Omega(\frac{N}{k^6})$.*

Proof. As in the proof of Corollary 5.3, we explain how each player can compute their respective outputs of a single unit of self-attention masked by $\Lambda^{w,g}$.

To compute the output corresponding to X_i, note that it is necessary to only know the embeddings corresponding to $X_{i-w}, X_{i-w+1}, \ldots, X_{i+w}$ and $X_g, X_{2g}, \ldots, X_{\lfloor N/g \rfloor g}$. Thus, player X^j can compute the outputs of all of their inputs $X^j = (X_{2(k-j)q+1}, \ldots, X_{2(k-j+1)q})$ given access to

$$X_{2(k-j)q+1-w}, \ldots, X_{2(k-j)q}, X_{2(k-j+1)q+1}, \ldots, X_{2(k-j+1)q+w},$$

as well as $X_g, X_{2g}, \ldots, X_{\lfloor N/g \rfloor g}$.

Therefore, the protocol can be simulated if each player X^j writes inputs

$$X_{2(k-j)q+1}, \ldots, X_{2(k-j)q+w}, X_{2(k-j+1)q-w+1}, \ldots, X_{2(k-j+1)q} \in \mathbb{R}^m,$$

in addition to all $X_i \in X^j$ such that $i \equiv 0 \pmod{g}$. This can be accomplished by a protocol where each player writes $s = O((w + \frac{N}{gk})mp)$ bits of information on the blackboard.

By repeating this protocol in parallel for every head and sequentially for every layer, T and $\mathrm{hop}_k(X)_N$ can be simulated, and hence the claim follows from Proposition E.4. □

E.4 Proofs of Section 5.4

Corollary 5.6. *Any $T \in \mathsf{MaskTransformer}^{N+N_{\mathrm{CoT}}}_{m,1,H}$ that computes $\mathrm{hop}_k(X)_N$ with N_{CoT} tokens of chain-of-thought requires either $N_{\mathrm{CoT}} \geq k$ or $mHp = \Omega(\frac{N}{k^6})$.*

Proof. We reduce to Proposition E.4. Consider some input $X \in \mathbb{R}^N$ partitioned into $X^1, \ldots, X^j$ as specified by the proof of Proposition E.4 with chain-of-thought X_{CoT} and $\mathrm{hop}_k(X)_N$ determined by some masked transformer T.[7] Suppose T has embeddings $(Q_h, K_h, V_h)_{h \in [H]}$ and output MLP ψ. We provide an $(N_{\mathrm{CoT}} + 1)$-round blackboard protocol to compute $\mathrm{hop}_k(X)_N$ from X.

Suppose in the rth round of the protocol, all players know $X_{\mathrm{CoT},1}, \ldots, X_{\mathrm{CoT},r-1}$ and aim to compute

$$\begin{aligned} T(X \circ X_{\mathrm{CoT}})_{N+r-1} &= \begin{cases} X_{\mathrm{CoT},r} & \text{if } r \leq N_{\mathrm{CoT}} \\ \mathrm{hop}_k(X)_N & \text{if } r = N_{\mathrm{CoT}} + 1 \end{cases} \\ &= \psi_{N+r-1}\left(X_{N+r-1} + \sum_{h=1}^{H} \frac{\sum_{i=1}^{N+r-1} \exp(Q^h_{N+r-1}(X_{N+r-1})^\mathsf{T} K^h_i(X_i)^\mathsf{T}) V^h_i(X_i)}{\sum_{i=1}^{N+r-1} \exp(Q^h_{N+r-1}(X_{N+r-1})^\mathsf{T} K^h_i(X_i))} \right). \end{aligned}$$

[7] We abuse notation to index $X_{N+i} = X_{\mathrm{CoT},i}$ and let $X_i \in X^j$ be true if $i \in \{2(k-j)q + 1, \ldots, w(k-j+1)q\}$.

If we let

$$\begin{aligned} S_{r,h,j} &= \sum_{X_i \in X^j} \exp(Q^h_{N+r-1}(X_{N+r-1})^\intercal K^h_i(X_i)^\intercal) V^h_i(X_i) \in \mathbb{R}^m, \\ S_{r,h,\mathrm{CoT}} &= \sum_{i=N+1}^{N+r-1} \exp(Q^h_{N+r-1}(X_{N+r-1})^\intercal K^h_i(X_i)^\intercal) V^h_i(X_i) \in \mathbb{R}^m, \\ Z_{r,h,j} &= \sum_{X_i \in X^j} \exp(Q^h_{N+r-1}(X_{N+r-1})^\intercal K^h_i(X_i)^\intercal) \in \mathbb{R}, \\ Z_{r,h,\mathrm{CoT}} &= \sum_{i=N+1}^{N+r-1} \exp(Q^h_{N+r-1}(X_{N+r-1})^\intercal K^h_i(X_i)^\intercal) \in \mathbb{R}, \end{aligned}$$

then we observe that

$$T(X \circ X_{\mathrm{CoT}})_{N+r-1} = \psi_{N+r-1}\left(X_{N+r-1} + \sum_{h=1}^{H} \frac{\sum_{j=1}^k S_{r,h,j} + S_{r,h,\mathrm{CoT}}}{\sum_{j=1}^k Z_{r,h,j} + Z_{r,h,\mathrm{CoT}}} \right).$$

Each player P_k computes $(S_{r,h,j}, Z_{r,h,j})_{h \in [H]}$ and writes them on the blackboard with $O(mHp)$-bit messages. Since $S_{r,h,\mathrm{CoT}}$ and $Z_{r,h,\mathrm{CoT}}$ are known by all players, every player can individually $T(X \circ X_{\mathrm{CoT}})_{N+r-1}$.

By induction, all players know $\mathrm{hop}_k(X)_N$ after $N_{\mathrm{CoT}} + 1$ rounds. The claim now follows from Proposition E.4. □

F Proofs of low-level attention constructions

F.1 Hardmax simulation proof of Appendix A.1

Lemma A.2. *Let $f \in \mathsf{Attn}^N_m$ be a self-attention unit with precision $p = \Theta(\log N)$ and embedding functions Q, K, V such that for some fixed $1 \geq \xi = N^{-O(1)}$ and every $X \in \mathbb{R}^{N \times m}$ and $i \in [N]$:*

$$A(X)_{i,i'} \leq \max_{i''} A(X)_{i,i''} - \xi, \ \forall i' \notin I_{\max}(A(X)_i),$$

where $A(X) = Q(X)K(X)^\intercal$. Then there exists a self-attention unit $f' \in \mathsf{Attn}^N_m$ with a valid p'-bit implementation with $p' = O(p)$ satisfying

$$f'(X) = \mathrm{hardmax}(A(X))V(X).$$

Proof. For some $p' = \Theta(p + \log \frac{1}{\xi})$ and $c = \Theta(\frac{p' + \zeta}{\xi} \cdot \log N)$ where ζ is as in Appendix A.1), let f' have query embedding $Q'(X) = cQ(X)$ and identical key K and value V embeddings as f. Therefore, by construction, these embeddings can be written with precision $p' = O(\ln(c) + p) = O(\log \frac{1}{\xi} + \log \log N + p) = O(p)$.

Let $\hat{f}'$ be a valid p'-bit implementation of f', meaning that the two $\|\hat{f}' - f'\|_\infty = O(1/2^{p+1})$ (thus $\hat{f}'$ rounds f' to p' bits of precision), and fix some X. We first show that the softmax matrix is sufficiently close to that of the hardmax and is also a valid p'-bit implementation of the hardmax. Without loss of generality, let $1 \in I_{\max}(A(X)_i)$. First, note that

$$\sum_{i' \notin I_{\max}(A(X)_i)} \exp(cA(X)_{i,i'}) \leq \frac{N}{\exp(c\xi)} \exp(cA(X)_{i,1}) = \frac{1}{N^{O(p'+\zeta)}} \exp(cA(X)_{i,1}).$$

Then,

$$\begin{aligned} |\mathrm{softmax}(cA(X))_{i,1} - \mathrm{hardmax}(A(X))_{i,1}| &= \frac{1}{|I_{\max}(A(X)_i)|} - \frac{\exp(cA(X)_{i,1})}{\sum_{i'=1}^N \exp(cA(X)_{i,i'})} \\ &\leq \frac{\sum_{i' \notin I_{\max}(A(X)_i)} \exp(cA(X)_{i,i'})}{|I_{\max}(A(X)_i)| \exp(cA(X)_{i,1})} = \frac{1}{N^{\Omega(p'+\zeta)}}. \end{aligned}$$

Likewise, for any $i'' \notin I_{\max}(A(X)_i)$:

$$|\mathrm{softmax}(cA(X))_{i,i''} - \mathrm{hardmax}(A(X))_{i,i''}| \leq \frac{\exp(cA(X)_{i,i''})}{\sum_{i'=1}^{N} \exp(cA(X)_{i,i'})} = \frac{1}{N^{\Omega(p'+\zeta)}}.$$

Therefore,

$$\|\mathrm{softmax}(cA(X))_i - \mathrm{hardmax}(cA(X))_i\|_2 \leq \sqrt{N} \cdot \max_{i''} |\mathrm{softmax}(cA(X))_{i,i''} - \mathrm{hardmax}(cA(X))_{i,i''}| = \frac{1}{N^{\Omega(p'+\zeta)}}.$$

We conclude that the approximation is sufficiently close, meaning it is $O(1/2^{p'})$, whereby it is exact after rounding:

$$\begin{aligned}
&\left\|\hat{f}'(X) - \mathrm{hardmax}(Q(X)K(X)^{\mathsf{T}})V(X)\right\|_\infty \\
&\quad \leq \left\|f'(X) - \mathrm{hardmax}(Q(X)K(X)^{\mathsf{T}})V(X)\right\|_\infty + \left\|\hat{f}'(X) - f'(X)\right\|_\infty \\
&\quad \leq \max_{i,j} \left|\mathrm{softmax}(cA(X))_i^{\mathsf{T}} V(X)_{\cdot,j} - \mathrm{hardmax}(A(X))_i^{\mathsf{T}} V(X)_{\cdot,j}\right| + O\left(\frac{1}{2^{p'}}\right) \\
&\quad \leq \max_{i,j} \left\|\mathrm{softmax}(cA(X))_i^{\mathsf{T}} - \mathrm{hardmax}(A(X))_i^{\mathsf{T}}\right\|_2 \left\|V(X)_{\cdot,j}\right\|_2 + O\left(\frac{1}{2^{p'}}\right) \\
&\quad \leq \frac{1}{N^{\Omega(p'+\zeta)}} \cdot \sqrt{N} \cdot N^{\zeta} + O\left(\frac{1}{2^{p'}}\right) = O\left(\frac{1}{2^{p'}}\right).
\end{aligned}$$

Therefore, $\hat{f}'$ is a valid p'-bit implementation of $\mathrm{hardmax}(Q(X)K(X)^{\mathsf{T}})V(X)$. □

F.2 Constructions for Appendix B.1

Proposition B.1. *For any $b \leq N$ and d, there exists a self-attention unit* $\mathrm{sparsePropagate}_{Q,d} \in \mathsf{Attn}_{m,p}^{N}$ *for $m = d + O(Q \log N)$ and $p = O(\log N)$, which, given any input X with $X_i = (z_i, S_i, \vec{0}) \in \mathbb{R}^d \times \binom{[N]}{\leq Q} \times \{0\}^{m-Q-d}$ such that $b_i = |\{S_j \ni i : j \in [N]\}| \leq Q$ for all i, has output* $\mathrm{sparsePropagate}_{Q,d}(X)$ *satisfying*

$$\mathrm{sparsePropagate}_{Q,d}(X)_i = \frac{1}{b_i} \sum_{S_j \ni i} z_j.$$

Proof. Following the proof of Theorem 2 of Sanford et al. (2023), there exist p-bit precision vectors $u_1, \dots, u_N \in \{\pm 1/\sqrt{m}\}^m$ and w_S with $w_S \leq 2\sqrt{Q}$ for all $S \in \binom{N}{\leq Q}$ such that

$$u_i^{\mathsf{T}} w_S = 1, \text{ for all } i \in S$$
$$u_i^{\mathsf{T}} w_S \leq \frac{1}{2}, \text{ for all } i \notin S.$$

We then design the embeddings of $\mathrm{sparsePropagate}_{Q,d}$ with

$$Q(X)_i = (u_i, 1),$$
$$K(X)_i = \begin{cases} (w_{S_i}, 0) & \text{if } i > 0, \\ (\vec{0}, \frac{3}{4}) & \text{if } i = 0, \end{cases}$$
$$V(X)_i = \begin{cases} z_i & \text{if } i > 0, \\ \vec{0} & \text{if } i = 0. \end{cases}$$

As a result,

$$\begin{aligned}
Q(X)_i^{\mathsf{T}} K(X)_{i'} &= 1 && \text{if } i \in S_{i'}, i' > 0, \\
Q(X)_i^{\mathsf{T}} K(X)_{i'} &\leq \frac{1}{2} && \text{if } i \notin S_{i'}, i' > 0, \\
Q(X)_i^{\mathsf{T}} K(X)_0 &= \frac{3}{4}.
\end{aligned}$$

Hence, the largest inner products for query i correspond to i' for all $S_{i'} \ni i$ if any exist, and 0 otherwise. There exists a margin of at least $\frac{1}{4}$ between the largest inner product in each row and all others. By applying Lemma A.2, we conclude that there exists a self attention unit f' with embedding dimension $p = \Theta(\log N)$ that computes

$$f'(X) = \mathrm{hardmax}(Q(X)K(X)^\intercal)V(X) = \mathrm{sparsePropagate}(X). \qquad \square$$

F.3 Constructions for Appendix B.2

Lemma B.4. *For any MPC protocol π with local memory s and q machines with n_{in}-word inputs, there exists a transformer* $\mathrm{init} \in \mathsf{Transformer}_{s,1,1,d_{\text{in}},d_{\text{out}}}^{n_{\text{in}},\max(n_{\text{in}},q)}$ *with $d_{\text{in}} = 1$ and $d_{\text{out}} = s$, which, given $\mathtt{Input} \in \mathbb{Z}_{2^p}^n$, has output satisfying* $\mathrm{init}(\mathtt{Input}) = \mathtt{MachineIn}^{(1)}$.

Proof. Let $M = \max(n_{\text{in}}, q)$ and $Q, K, V : \mathbb{Z}_{2^p}^M \to \mathbb{R}^{M \times s}$ be the query, key, and value embeddings of the attention unit f in init, and let $\psi : \mathbb{R}^{M \times s} \to \mathbb{Z}_{2^p}^s \times [N]$ be its output MLP. Let $q_{\text{in}} = \left\lceil \frac{n_{\text{in}}}{s} \right\rceil$ denote the number of machines used to store the inputs.

Let $\mathtt{Dest}_{i'} = \left\lceil \frac{i'}{s} \right\rceil \in [q_{\text{in}}]$ denote the machine that stores the input token index $i' \in [n_{\text{in}}]$ in the MPC protocol, and let

$$\mathtt{Rcvd}_i = \{(s-1)i + 1, \dots, \min(si, n_{\text{in}})\}$$

denote the set of all input tokens indices belonging to $\mathtt{MachineIn}_i^{(1)}$ for machine $i \in [q_{\text{in}}]$.

For each machine $i \in [q_{\text{in}}]$, we define the query embedding as

$$Q(\mathtt{Input})_i = \left(\cos\left(\frac{2\pi i}{M}\right), \sin\left(\frac{2\pi i}{M}\right), \dots, \cos\left(\frac{2\pi i}{M}\right), \sin\left(\frac{2\pi i}{M}\right)\right).$$

Likewise, for each token index $i' \in [n_{\text{in}}]$, the key and value vectors are

$$K(\mathtt{Input})_{i',(2\iota-1,2\iota)} = \begin{cases} \left(\cos\left(\frac{2\pi \cdot \mathtt{Dest}_{i'}}{M}\right), \sin\left(\frac{2\pi \cdot \mathtt{Dest}_{i'}}{M}\right)\right) & \text{if } i' \leq n_{\text{in}},\ i' \equiv \iota \pmod{s}, \\ (0, 0) & \text{otherwise,} \end{cases}$$

$$V(\mathtt{Input})_{i',(2\iota-1,2\iota)} = \begin{cases} (\mathtt{Input}_{i'}, i') & \text{if } i' \leq n_{\text{in}},\ i' \equiv \iota \pmod{s}, \\ (0, i') & \text{otherwise.} \end{cases}$$

These definitions guarantee that large inner products only occur between machine queries $Q(\mathtt{Input})_i$ and tokens keys $K(\mathtt{Input})_{i'}$ when $\mathtt{Input}_{i'}$ is allocated to $\mathtt{MachineIn}_i^{(1)}$. That is,

$$\begin{aligned} Q(\mathtt{Input})_i^\intercal K(\mathtt{Input})_{i'} &= 1, && \text{if } i' \in \mathtt{Rcvd}_i \\ Q(\mathtt{Input})_i^\intercal K(\mathtt{Input})_{i'} &\leq 1 - \Omega\left(\frac{1}{M^2}\right), && \text{otherwise.} \end{aligned}$$

By applying Lemma A.2 with $\xi = \Omega(\frac{1}{N^2})$, there exists some self-attention unit f' such that

$$f'(\mathtt{Input})_i = \mathrm{hardmax}(Q(\mathtt{Input})K(\mathtt{Input})^\intercal) = \frac{(\mathtt{Input}_{i'}, i')_{i' \in \mathtt{Rcvd}_i}}{|\mathtt{Rcvd}_i|}.$$

A proper choice of ψ and an invocation of the definition of $\mathtt{MachineIn}^{(1)}$ ensures that $\mathrm{init}(\mathtt{Input})_i = \psi(f(\mathtt{Input}))_i = \mathtt{MachineIn}_i^{(1)}$. $\square$

Lemma B.6. *For any R-round MPC protocol π with local memory s and q machines with n_{out}-word output, there exists a transformer* $\mathrm{final} \in \mathsf{Transformer}_{s,1,1,d_{\text{in}},d_{\text{out}}}^{q,\max(n_{\text{out}},q)}$ *for $d_{\text{in}} = s$ and $d_{\text{out}} = 1$, which, given input $X = \mathtt{MachineIn}^{(R)}$, has output* $\mathrm{final}(X)$ *with* $\mathrm{final}(X)_{i,1} = \mathtt{Output}_i \in \mathbb{Z}_{2^p}$.

Proof. This argument inverts that of Lemma B.4, after applying the $\texttt{Local}_R$ to transform $\texttt{MachineIn}^{(R)}$ to $\texttt{MachineOut}^{(R)}$. Let $Q, K, V : \mathbb{Z}_{2^p}^M \to \mathbb{R}^{M \times s}$ be the query, key, and value embeddings of the only attention unit f in final, and let $\psi : \mathbb{R}^{M \times s} \to \mathbb{Z}_{2^p}^s \times [N]$ be its output MLP. Let $q_{\mathrm{out}} = \lceil \frac{n_{\mathrm{out}}}{s} \rceil$ denote the number of machines storing relevant information for the output of the MPC protocol.

For each machine $i' \in [q_{\mathrm{out}}]$, let

$$\texttt{Sent}_{i'} = \{(s-1)i' + 1, \dots, \min(si', n_{\mathrm{out}})\}$$

denote the set of all token indices receiving its output. Likewise, for each token index $i \in [n_{\mathrm{out}}]$, let $\texttt{Src}_i = \lceil i/s \rceil$ be the machine containing its relevant token. We define $Q = Q' \circ \texttt{Local}_R, K = K' \circ \texttt{Local}_R, V = V' \circ \texttt{Local}_R$ as follows.

$$\begin{aligned}
Q'(\texttt{MachineOut}^{(R)})_{i,(2\iota-1,2\iota)} &= \begin{cases} \left(\cos\left(\frac{2\pi \lfloor \texttt{Src}_i \rfloor}{M}\right), \sin\left(\frac{2\pi \lfloor \texttt{Src}_i \rfloor}{M}\right)\right) & \text{if } i \leq n_{\mathrm{out}},\ i \equiv \iota \pmod{s} \\ (0,0) & \text{otherwise.} \end{cases} \\
K'(\texttt{MachineOut}^{(R)})_{i'} &= \left(\cos\left(\frac{2\pi i'}{M}\right), \sin\left(\frac{2\pi i'}{M}\right), \dots, \cos\left(\frac{2\pi i'}{M}\right), \sin\left(\frac{2\pi i'}{M}\right)\right). \\
V'(\texttt{MachineOut}^{(R)})_{i'} &= \texttt{MsgOut}_{i'}^{(R)}.
\end{aligned}$$

Applying Lemma A.2 as before yields

$$f(\texttt{MachineIn}^{(R)})_i = \begin{cases} \texttt{MachineOut}_{i'}^{(R)} & \text{if } i \in \texttt{Sent}_{i'}, \\ 0 & \text{otherwise.} \end{cases}$$

A properly chosen ψ ensures that $\mathrm{final}(\texttt{MachineIn}^{(R)})_i = \psi(f(\texttt{MachineIn}^{(R)}))_i = \texttt{Output}_i$. □

F.4 Constructions for Appendix D.1

Lemma D.1. *For some $m \geq d+2$, $\tau : [N] \times \mathbb{R}^m \to [N]$, and $\rho : \mathbb{R}^m \to \mathbb{R}^d$, there exists an attention head $\mathrm{lookUp}_{\tau,\rho} \in \mathsf{MaskAttn}_m^N$ with precision $p = O(\log N)$ and $m \geq d+2$ satisfying $\mathrm{lookUp}_{\tau,\rho}(X)_{i,:d} = \rho(X_{\tau(i,X_i)})$.*

Proof. We let $V(X_i) = (\rho(X_i), \vec{0})$ and define sinusoidal embeddings Q and K with

$$\begin{aligned}
Q(X)_i &= \left(\cos\left(\frac{2\pi\tau(i,X_i)}{N}\right), \sin\left(\frac{2\pi\tau(i,X_i)}{N}\right), \vec{0}\right), \\
K(X)_i &= \left(\cos\left(\frac{2\pi i}{N}\right), \sin\left(\frac{2\pi i)}{N}\right), \vec{0}\right).
\end{aligned}$$

Note that

$$\begin{aligned}
Q(X)_i^\mathsf{T} K(X)_{i'} &= 1, && \text{if } \tau(i,X_i) = i', \\
Q(X)_i^\mathsf{T} K(X)_{i'} &\leq \cos\left(\frac{2\pi}{N}\right) = 1 - \Omega\left(\frac{1}{N^2}\right), && \text{otherwise.}
\end{aligned}$$

By applying Lemma A.2 with $\xi = \Omega(\frac{1}{N^2})$, we conclude that a satisfactory self-attention unit exists. □

Lemma D.2. *For any finite alphabet Σ, $m \geq d+2$, $\mu_1, \mu_2 : \mathbb{R}^m \to \Sigma$, and $\rho : \mathbb{R}^m \to \mathbb{R}^d$, there exists an attention head $\mathrm{lastOccurrence}_{\mu,\rho} \in \mathsf{MaskAttn}_m^N$ with precision $p = O(\log(N\,|\Sigma|))$ such that,*

$$\mathrm{lastOccurrence}(X)_{i,:d} = \begin{cases} \rho(\vec{0}) & \text{if } \forall\, i' < i : \mu_1(X_{i'}) \neq \mu_2(X_i), \\ \rho(X_{i'}) & \text{if } i' = \max\{i' < i : \mu_1(X_{i'}) = \mu_2(X_i)\}. \end{cases}$$

Proof. Let $N' = N|\Sigma|$. We define token embeddings as follows, including start token "dummy embeddings" as discussed in Appendix A.1.

$$
\begin{aligned}
Q(X)_i &= \left(\cos\left(\frac{2\pi(N\mu_2(X_i)+i)}{N|\Sigma|}\right), \sin\left(\frac{2\pi(N\mu_2(X_i)+i)}{N|\Sigma|}\right), 1, \vec{0}\right), \\
K(X)_i &= \left(\cos\left(\frac{2\pi(N\mu_1(X_i)+i)}{N|\Sigma|}\right), \sin\left(\frac{2\pi(N\mu_1(X_i)+i)}{N|\Sigma|}\right), 0, \vec{0}\right), \\
K(X)_0 &= \left(0, 0, \cos\left(\frac{2\pi(N-\frac{1}{2})}{N|\Sigma|}\right), \vec{0}\right), \\
V(X)_i &= (\rho(X_i), \vec{0}), \\
V(X)_0 &= \vec{0}.
\end{aligned}
$$

Taken together, these embeddings provide the following characterization of the inner products (with causal masking matrix Γ):

$$
\begin{aligned}
Q(X)_0^\top K(X)_{i'} + \Gamma_{i,i'} &= \cos\left(\frac{2\pi(i-i')}{N|\Sigma|}\right) && \text{if } i \geq i' > 0,\ \mu_1(X_{i'}) = \mu_2(X_i), \\
Q(X)_i^\top K(X)_{i'} + \Gamma_{i,i'} &\leq \cos\left(\frac{2\pi}{N}\right) && \text{if } i \geq i' > 0,\ \mu_1(X_{i'}) \neq \mu_2(X_i), \\
Q(X)_i^\top K(X)_{i'} + \Gamma_{i,i'} &= -\infty && \text{if } i < i', \\
Q(X)_i^\top K(X)_i + \Gamma_{i,0} &= \cos\left(\frac{2\pi(N-\frac{1}{2})}{N|\Sigma|}\right).
\end{aligned}
$$

As a result, the largest inner product $Q(X)_i^\top K(X)_{i'}$ for some i is the largest i' with $\mu_1(X_{i'}) = \mu_2(X_i)$ if one exists and $i' = 0$ otherwise. Furthermore, there exists a margin of $\Omega(\frac{1}{N^2|\Sigma|^2})$ between this inner product and all others. We conclude by applying Lemma A.2. □

G Further Empirical Analysis of k-Hop Induction Heads

This appendix presents in-depth explanations of the empirical results of Section 4.2, along with further experiments. Taken together, these results suggest that the relationship between the number of hops k and the depth L of transformers trained on the task is well-characterized by the representational thresholds of Theorem 4.2 and Corollary 4.3; that the construction described in the proof of Theorem 4.2 is attainable by trained models; and deep models likely exhibit an inductive bias that favors compositional learning rules in the finite sample regime.

We define our experimental methodology precisely in Appendix G.1 and provide supporting evidence for our claims in the subsequent sections.

Exponential Powers of Depth. Our principal empirical claim is that incrementing the depth L of a transformer exponentially increases the model's capabilities to learn k-hop induction heads tasks. We explore this claim primarily in Appendix G.2, where we compare this empirical claim with the relevant theoretical results (Theorem 4.2 and Corollary 4.3), which suggest a similar dependence. We further study the impacts of increasing the embedding dimension m of the transformer in Appendix G.3 and find that doubling the width is roughly equivalent in performance to incrementing the depth by one.

Empirical Claim G.1. A transformer $T \in \mathsf{MaskTransformer}_{m,L,H}^N$ trained with Adam to solve hop_k has small token-wise classification error if $L\log(m) = \Omega(\log k)$ and large error if $L \log m = O(\log k)$.

Mechanistic Alignment with Theoretical Construction. We further demonstrate the empirical salience of our theoretical construction by conducting a study of the interpretability of learned transformers in Appendix G.4. This investigation reveals that the attention matrices of sufficiently deep transformers exhibit an implementation of a circuit that relies on the same "doubling" principle of the construction in the proof of Theorem 4.2. The resulting circuit is comprised of the same intermediate products that are used in that hop_k construction.

Empirical Claim G.2. The outputs of individual attention matrices of a transformer $T \in \mathsf{MaskTransformer}_{m,L,H}^N$ trained with Adam to solve hop_k with $L = \Omega(\log k)$ and evaluated on input $X \in \Sigma^N$ (i) correspond to the find_X^j intermediate products of the Theorem 4.2 construction and (ii) demonstrate a "doubling" phenomenon where the each head layer ℓ corresponds to find_X^j for some $j = O(2^\ell)$.

Beneficial Inductive Biases of Depth. While most of our experiments belong to the "infinite-sample" regime where new samples are randomly generated on each training step, we also evaluate our models in two finite-sample regimes in Appendix G.5. We find that a small number of samples is sufficient to approach the performance of the infinite-sample regime. When the amount of training data is small, we find that deeper models perform better than shallower models, possibly due to an inductive bias that favors compositional hypotheses.

Empirical Claim G.3. hop_k can be learned in a sample-efficient manner by transformers $T \in \mathsf{MaskTransformer}_{m,L,H}^N$ trained with Adam with $L = \Omega(\log k)$. If T overfits to hop_k tasks for some k, then increasing the depth L while holding k fixed leads superior performance.

The experiments detailed here were conducted under limited computational resources. The authors are interested in future work that would evaluate whether these scaling rules persist on larger architectures and more complex tasks.

G.1 Experimental Details

Task Details. We study a multi-task variant of k-hop induction heads that predicts $\mathrm{hop}_k(X) = (0, \mathrm{hop}_k(X'))$ from input $X = (k, X')$ for $k \in \{0, 1, \dots, k_{\max}\}$[8] and $X' \in \Sigma^{N-1}$. We refer to this task as *multi-hop* and provide the task hyper-parameters in Table 1.

Hyperparameter	Value
Context length N	100
Alphabet size $\lvert\Sigma\rvert$	4
Max hops $k_{\max}$	16

Table 1. Multi-hop task hyper-parameters

We define the distribution $\mathcal{D}_{\mathrm{multi-hop}}$ over labeled samples for the multi-hop task and $\mathcal{D}_{\mathcal{X}}$ over input sequences $X \in \Sigma^{N-1}$. We draw a labeled sample $(X, \mathrm{hop}_k(X)) \sim \mathcal{D}_{\mathrm{multi-hop}}$ by independently sampling $k \sim \mathrm{Unif}(\{0, 1, \dots, k_{\max}\})$ and $X' \sim \mathcal{D}_{\mathcal{X}}$. Input sequences $X' \sim \mathcal{D}_{\mathcal{X}}$ are drawn uniformly from inputs *with no repeating elements*. That is, we sample $X_1' \sim \mathrm{Unif}(\Sigma)$ and each $X_{j+1}' \sim \mathrm{Unif}(\Sigma \setminus \left\{X_j'\right\})$. For each $k \in [k_{\max}]$, let $\mathcal{D}_{\mathrm{hop}_k}$ denote the conditional distribution $((k', X'), (0, \mathrm{hop}_{k'}(X'))) \sim \mathcal{D}_{\mathrm{multi-hop}} \mid (k = k')$. Also, let $\mathrm{dom}(\mathrm{hop}_k) = \{(k, X') : \Pr\left[X' \sim \mathcal{D}_{\mathcal{X}}\right] > 0\}$.

For $\overline{\Sigma} := \Sigma \cup [k_{\max}]$, we define the n-sample *empirical token-wise classification error* of a transformer $T : \overline{\Sigma}^N \to \overline{\Sigma}^N$ on a task hop_k as

$$\mathtt{err}_k^n(T) = \frac{1}{n} \sum_{\iota=1}^{n} \frac{1}{\lvert \{i : \mathrm{hop}_k(X^\iota)_i \neq \perp\} \rvert} \sum_{i=1}^{N} \mathbb{1}\left\{T(X^\iota)_i \neq \mathrm{hop}_k(X^\iota)_i \neq \perp\right\},$$

for iid samples $(X^1, \mathrm{hop}_k(X^1)), \dots, (X^n, \mathrm{hop}_k(X^n)) \sim \mathcal{D}_{\mathrm{hop}_k}$. We ignore null $\perp$ outputs of hop_k when no k-hop induction head exists in order to avoid inadvertently over-estimating the performance of transformers on large k tasks, which have a large fraction of null outputs.

Training Details. We trained a variety of causally-masked GPT-2 transformers (Radford et al., 2019) from HuggingFace to solve the multi-hop task. The model has an absolute positional encoding.

The transformers are trained with Adam (Kingma & Ba, 2014) on the cross-entropy loss. In the infinite-sample regime, we draw 32 new iid samples from $\mathcal{D}_{\mathrm{multi-hop}}$ on each training step. Otherwise, n_{train} samples are drawn before training commences and all samples are rotated through batches, before repeating. We use the hyper-parameters in Table 2 to train all of the models identified in Table 3.

Computational Resources. All experiments were run on a 2021 Macbook Pro with an M1 chip.

[8] The task hop_0 is simply the identity mapping: $\mathrm{hop}_0(X') = X'$.

Hyperparameter	Value
Embedding dimension m	$\{128, 256\}$
Depth L	$\{2, 3, 4, 5, 6\}$
Number of heads H	$\{4, 8\}$
Vocabulary size	30
Activation function	GeLU
Layer norm ϵ	10^{-5}
Training samples n_{train}	$\left\{10^3, 3 \cdot 10^3, \infty\right\}$
Learning rate	10^{-4}
Training steps	10^5
Batch size	32

Table 2. Model and training hyper-parameters

Identifier	Heads H	Embedding dimension m	Depth L	Training samples n_{train}	Total parameters
$T_{4,2}^{\infty}$	4	128	2	∞	413,440
$T_{4,3}^{\infty}$	4	128	3	∞	611,712
$T_{4,4}^{\infty}$	4	128	4	∞	809,984
$T_{4,5}^{\infty}$	4	128	5	∞	1,008,256
$T_{4,6}^{\infty}$	4	128	6	∞	1,206,528
$T_{8,2}^{\infty}$	8	256	2	∞	1,613,312
$T_{8,3}^{\infty}$	8	256	3	∞	2,403,072
$T_{8,4}^{\infty}$	8	256	4	∞	3,192,832
$T_{8,5}^{\infty}$	8	256	5	∞	3,982,592
$T_{8,6}^{\infty}$	8	256	6	∞	4,772,352
$T_{4,2}^{3000}$	4	128	2	3000	413,440
$T_{4,3}^{3000}$	4	128	3	3000	611,712
$T_{4,4}^{3000}$	4	128	4	3000	809,984
$T_{4,5}^{3000}$	4	128	5	3000	1,008,256
$T_{4,6}^{3000}$	4	128	6	3000	1,206,528
$T_{4,2}^{1000}$	4	128	2	1000	413,440
$T_{4,3}^{1000}$	4	128	3	1000	611,712
$T_{4,4}^{1000}$	4	128	4	1000	809,984
$T_{4,5}^{1000}$	4	128	5	1000	1,008,256
$T_{4,6}^{1000}$	4	128	6	1000	1,206,528

Table 3. Hyper-parameters of all $\mathsf{MaskTransformer}_{m,L,H}^{N}$ trained for the empirical analysis.

G.2 Exponential Increases in k-Hop Capacity with Depth (Empirical Claim G.1; Figures 6 to 8)

We visualize the relationship between the depth L of a transformer and the largest k such that $\texttt{err}_k^n(T)$ is small in Figure 6, Figure 7, and Figure 8. We exhibit the relationship in its simplest form by considering transformers with heads $H = 4$, embedding dimension $m = 128$, and new training samples on every epoch. The figures provide alternate views of $\texttt{err}_k^n(T_{4,L}^{\infty})$ for each $L \in \{2, 3, 4, 5, 6\}$ with $n = 100$ samples for each $k \in [k_{\max}]$.

Together, these plots illustrate a sharp phase transition when $D = \lfloor \log_2 k \rfloor + 2$, which identically matches the depth scaling in Theorem 4.2. Increasing the depth of a transformer by one approximately doubles the number of values $k \in [k_{\max}]$ with bounded error. For instance, following the theoretical and empirical intuition of (Bietti et al., 2023), the depth $L = 2$ transformer $T_{4,2}^{\infty}$ succeeds in solving the standard induction heads task, but attains at least 10% error on all other tasks. Likewise, a depth $L = 3$ model has error bounded by 1% for $k \in \{1, 2\}$, which increases rapidly for larger values of k.

This doubling phenomenon suggests that simple compositional tasks with a larger number of compositions than the depth of the model are easily learnable if the model can employ a doubling trick, similar to the one used in the proof of Theorem 4.2.

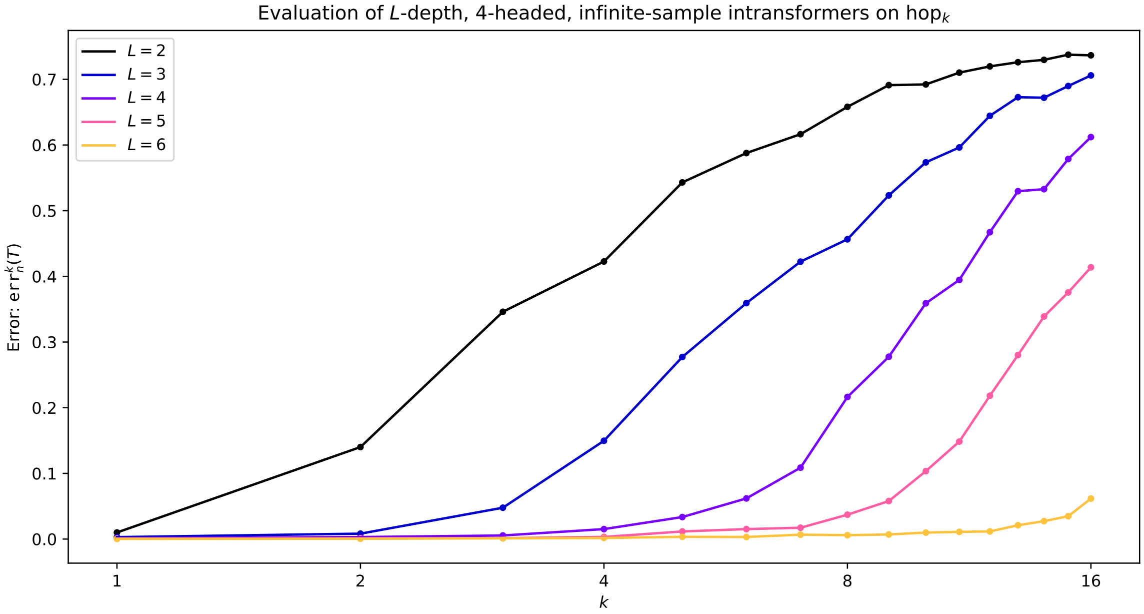


Figure 6. Zoomed in version of Figure 2. Evaluation of transformers $\mathtt{err}_k^n(T_{4,L}^{\infty})$ with depths $L \in \{2, 3, 4, 5, 6\}$, heads $H = 4$, and embedding dimension $m = 128$ trained on the multi-hop task. This figure plots $\mathtt{err}_k^n(T_{4,L}^{\infty})$ on $n = 100$ samples as a function of k for each choice of L.

This relationship between compositionality and depth reflects the results of Zhang et al. (2023), where the learnable task complexity also scales super-linearly in depth.

Given the lower bounds of Corollary 4.3, one may ask why models with depth $L < \lfloor \log_2 k \rfloor$ achieve non-trivial success on hop_k tasks that cannot be represented in a compositional manner. There are several relevant explanations:

1. In these experiments, the embedding dimension $m = 128$ is actually larger than the context $N = 100$, which may enable the model to memorize more of its preceding samples and offload logical work to the MLP, rather than executing a pointer-doubling strategy. While practical models regularly have the opposite (and our theoretical results are oriented around that parametric scaling), we used a larger m than is necessary for representational purpose to improve the optimization landscape and speed convergence.

2. This is made further plausible by the small alphabet size $|\Sigma|$ and randomly drawn sequences X', which place effective bounds on how much look-back from each token i is necessary to compute $\mathrm{hop}_k(X)_i$.

Nonetheless, these results provide strong support that models are substantially easier to train to low classification error in the regime where the depth is sufficient to implement a pointer-doubling construction. In the following subsection, we further investigate this phenomenon by examining the intermediate attention matrices produced by trained models.

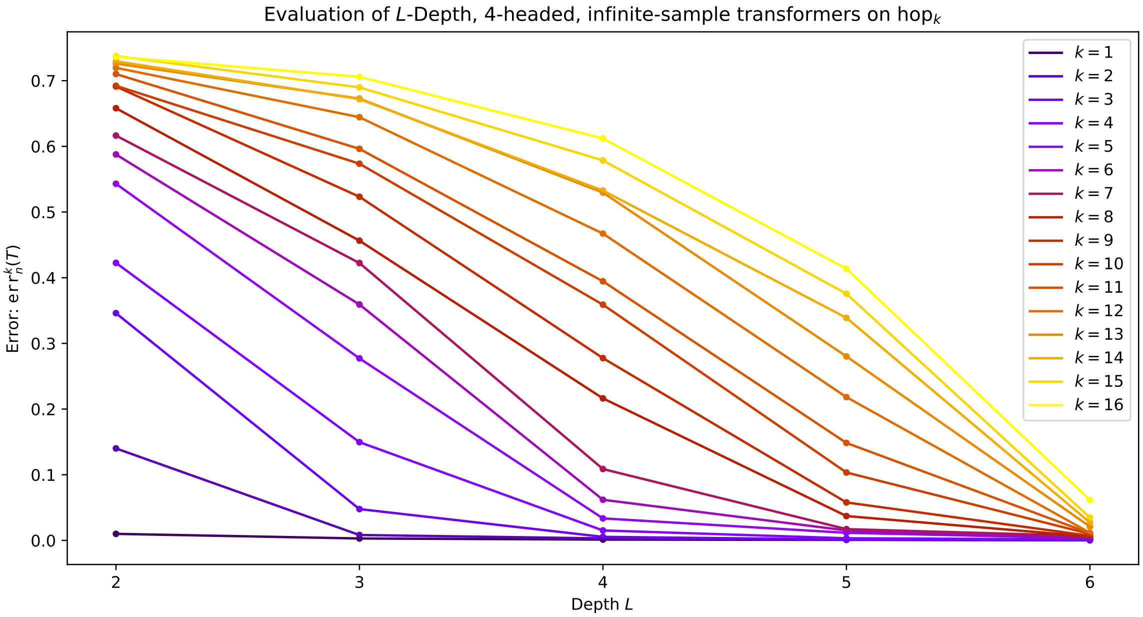
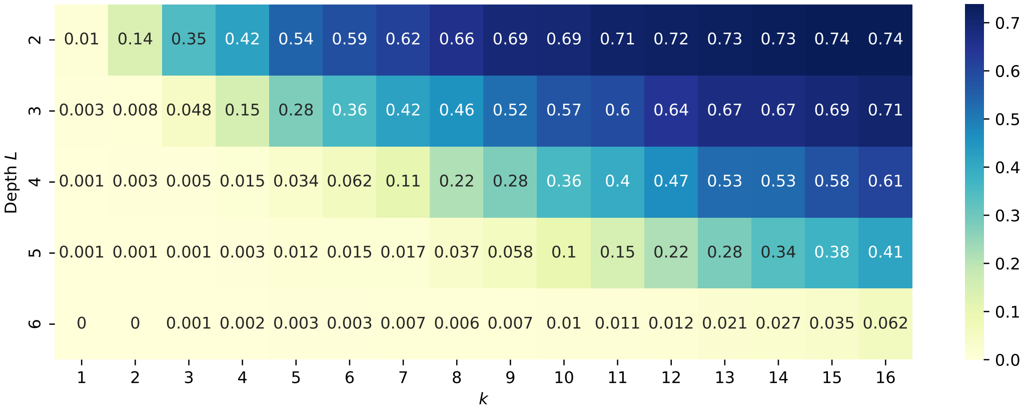


Figure 7. Alternate view of Figure 6 including $\texttt{err}_k^n(T_{4,L}^\infty)$ plotted as a function of L for each k.



Depth L \ k	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
2	0.01	0.14	0.35	0.42	0.54	0.59	0.62	0.66	0.69	0.69	0.71	0.72	0.73	0.73	0.74	0.74
3	0.003	0.008	0.048	0.15	0.28	0.36	0.42	0.46	0.52	0.57	0.6	0.64	0.67	0.67	0.69	0.71
4	0.001	0.003	0.005	0.015	0.034	0.062	0.11	0.22	0.28	0.36	0.4	0.47	0.53	0.53	0.58	0.61
5	0.001	0.001	0.001	0.003	0.012	0.015	0.017	0.037	0.058	0.1	0.15	0.22	0.28	0.34	0.38	0.41
6	0	0	0.001	0.002	0.003	0.003	0.007	0.006	0.007	0.01	0.011	0.012	0.021	0.027	0.035	0.062

Figure 8. Alternate views of Figure 6 including $\texttt{err}_k^n(T_{4,L}^\infty)$ as a table with one cell for each (L, k) pair.

G.3 Width Variation (Empirical Claim G.1; Figure 9)

While the primary focus of these empirical results and the paper as a whole is on the role of depth in the ability of transformer to learn parallelizable and compositional tasks, we also aim to understand the interplay of depth and width in learning the multi-hop task. Here, we contrast the previous transformers $T^{\infty}_{4,L}$ with models $T^{\infty}_{8,L}$ that have more heads ($H = 8$) and larger embedding dimensions ($m = 256$). We plot the classification errors of all 10 architectures over 16 hop_k sub-tasks in Figure 9.

Here, we observe a rough correspondence in performance between the transformers $T^{\infty}_{H,L}$ and $T_{2H,L-1}$ and the same doubling phenomenon as is evident models with $H = 4$ heads. That is, while increasing the width improves the classification error of learned models, it does so in a far less parameter-efficient manner than incrementing the depth. As mentioned before, the relative success of wide and shallow transformers is likely contingent on the relatively short context length N and alphabet size $|\Sigma|$. However, these results still suggest an important role for wider models to play beyond representational capabilities of transformers.

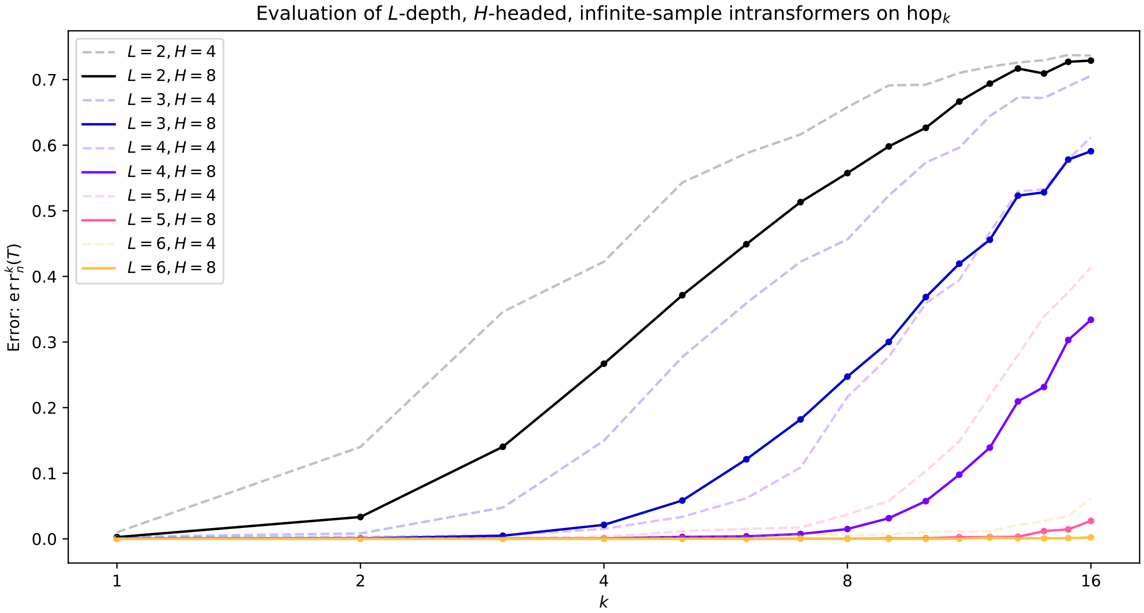


Figure 9. Comparison between the errors $\mathrm{err}_k^n(T^{\infty}_{H,L})$ of transformers with embedding dimension and heads $(m, H) = (4, 128)$ (dashed line, same plots as Figure 6) and $(m, H) = (8, 256)$ (solid line) trained on the multi-hop task, evaluated on $n = 100$ samples per hop_k task.

G.4 Mechanistic Alignment with Theoretical Construction (Empirical Claim G.2, Figures 10 to 15)

We use standard attention-based interpretability techniques to better understand what particular logical circuits are implemented by transformers trained to solve the multi-hop task. By qualitatively inspecting the attention matrices produced by trained models and by measuring the alignment between those inner products and partial solutions find^j of hop_k, we uncover a striking correspondence between the behaviors of the trained models and the transformer construction designed in the proof of Theorem 4.2. We further observe that trained transformers with high accuracy have "decisive" self-attention units with particularly strong correlations to some find^j intermediate, while poorly performing models have less predictable attention activations.

For a fixed trained model $T \in \mathsf{Transformer}_{m,L,H}^N$, we let $A^{\ell,h}[T](X)$ represent the output of the hth self-self attention matrix in the ℓth layer for $h \in [H]$ and $\ell \in [L]$, evaluated at some input $X \in \mathrm{dom}(\mathrm{hop}_k)$. That is, we let

$$A^{\ell,h}[T](X) = \mathrm{softmax}\left(Q^{\ell,h}(X^{\ell-1})K^{\ell,h}(X^{\ell-1})^\mathsf{T} + \Gamma\right) \in \mathbb{R}^{N \times N},$$

where $X^{\ell-1}$ is the intermediate state representing the output of layer $\ell - 1$ of T on input X and Γ is the causal masking matrix. Each row i in the matrix represents the coefficients of the convex combination of value vectors affiliated with each query, which can be used as a signifier of which embeddings i receives information from.

Visualization of find^j Alignment for hop_{16} and Depth $L = 6$ (Figure 10). The outputs of self-attention matrices are often highly structured matrices that reveal which relationships between tokens are encoded and how information is shared within the model (Li & McClelland, 2022; Clark et al., 2019; Rogers et al., 2021). We plot several self-attention matrices associated with a depth $L = 6$, heads $H = 4$ transformer trained in the infinite-sample regime and evaluated on a single sample $X \in \mathrm{dom}(\mathrm{hop}_{16})$ in Figure 10.

By looking at the six self-attention matrices, one can infer that all heads are "decisive" and obtain nearly all of their relevant information from a single value embedding, rather than averages of a large number of embeddings. The top-left self-attention matrix, which belongs to the first self-attention head, clearly associates elements with their predecessors, which is identical the to the function of our lookUp attention head in the first layer of the hop_k construction of Theorem 4.2.

While the roles of the other heads are not immediately obvious, they can be understood by overlaying colored matrices with non-zero cells at $(i, \mathrm{find}_X^j(i))$ for some $j \le k$. For instance, the top-right attention matrix in layer $\ell = 2$ corresponds almost exactly with find_X^1 (as suggested by the second-layer of our construction), and the others are closely associated with find_X^1, find_X^2, find_X^3, and find_X^8 for layers $\ell = 3, 4, 5, 6$ respectively. This is a remarkably close correspondence to our construction, which includes a self-attention matrix in the ℓth layer whose activations correspond to $\mathrm{find}_X^{2^{\ell-2}}$.

While not conclusive, this experiment suggests a strong alignment between the behaviors of this particular transformer and our theoretical construction. This suggests a high likelihood that the transformer successfully learns to solve hop_{16} by employing a pointer-doubling primitive. However, these results apply to only a single model, a single task, and a single input; in the subsequent section, we generalize this interpretability analysis.

Alignment between Attention Jeads and find^j for a Single hop_k Sub-Task (Figures 11 to 13). To broaden and quantify the analysis of the previous section, we measure the extent to which each self-attention head mimics the functionality of find^j, which are partial computations of hop_k that are employed in the proof of Theorem 4.2. We use cell-wise matrix inner products to quantify the strength of correlation between a self-attention matrix and a fixed function potentially relevant to interpretability.

For two matrices $A, B \in \mathbb{R}^{N \times N}$, let

$$\langle A, B \rangle = \frac{\|A \odot B\|_F^2}{\|A\|_F \|B\|_F}$$

be their normalized element-wise inner-product, where $\|\cdot\|_F$ is the Frobenius norm and $\odot$ denotes element-wise multiplication. For some function $g : [N] \to \{0\} \cup [N]$, we let $\langle g, B \rangle := \langle A^g, B \rangle$, where

$$A_{i,j}^g = \begin{cases} 1 & \text{if } g(j) = i, \\ 0 & \text{otherwise.} \end{cases}$$

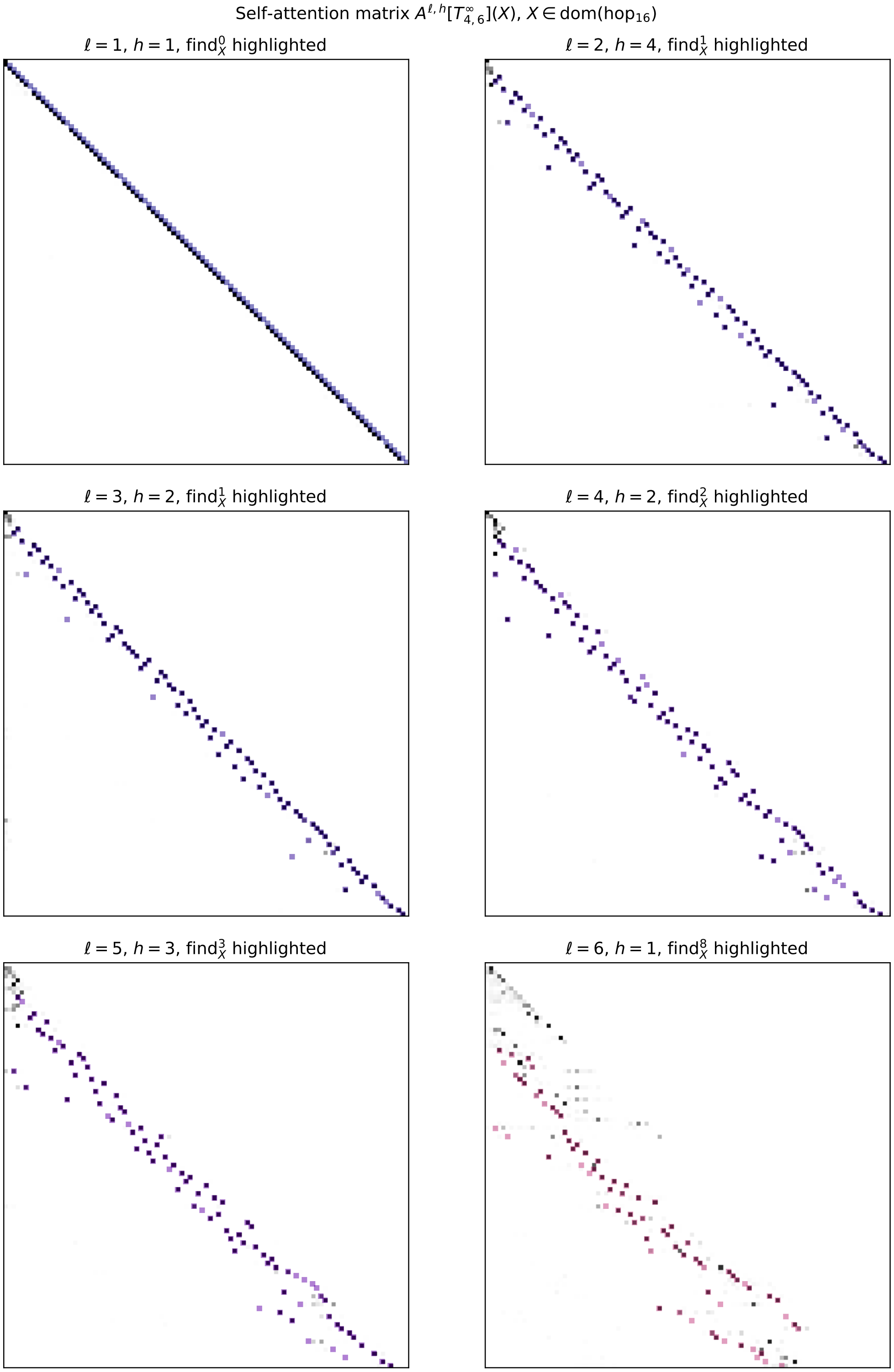


Figure 10. The outputs of several internal self-attention matrices $A^{\ell,h}[T^{\infty}_{4,6}](X) \in \mathbb{R}^{100\times 100}$ of a trained multi-task transformer of depth $D = 6$ evaluated on a single sample $X \sim \mathcal{D}_{\mathrm{hop}_{16}}$ are plotted in grayscale. In each cell, the matrix with non-zero entries $(\mathrm{find}^j_X(i), i)_{i\in[N]}$ for some j is included in transparent color to visualize the function of each self-attention unit.

We use this notation to analyze experimentally how closely the self-attention matrices $A^{\ell,h}$ encode the intermediate products of the proof of Theorem 4.2, find_X^j. For n iid samples $X^1, \dots, X^n \in\sim \mathcal{D}_{\mathrm{hop}_k}$, let

$$\left\langle A^{\ell,h}, \mathrm{find}^j \right\rangle_{n,k} := \frac{1}{n} \sum_{\iota=1}^{n} \left\langle \mathrm{find}_{X^\iota}^j, A^{\ell,h}(X^\iota) \right\rangle.$$

Due to the non-negativity of $A^{\ell,h}$ and find^j, $\left\langle A^{\ell,h}, \mathrm{find}^j \right\rangle_{n,k} \in [0,1]$, and $\left\langle A^{\ell,h}, \mathrm{find}^j \right\rangle_{n,k} = 1$ only if $\forall \iota \in [n]$:

$$A^{\ell,h}(X^\iota)_{i,i'} = 1 \iff \mathrm{find}_{X^\iota}^j(i) = i'.$$

These inner products make it possible to visualize the strength of correlations of all heads in a particular model $T \in \mathsf{MaskTransformer}_{m,L,H}^N$ with all target functions find^j on a collection of random samples drawn from some $\mathcal{D}_{\mathrm{hop}_k}$. Figure 11 visualizes the functionality of all attention units in the 4-layer, 4-head transformer $T_{4,4}^\infty$ when evaluated on the sub-task hop_4. The figure gives several clues about how hop_4 is successfully computed by the trained model: the second layer and third layer both utilize find^1 to determined find^2 jointly by the end of the third layer. The fourth layer uses the ability to create a stable find^2 construction to obtain find^4 and hence hop_4.

This plot also indicates the relative stability of this circuit interpretation of the procedure: a large number of heads are very strongly correlated with find^1 or find^2 across the 10 samples, which indicates they are likely utilized consistently to compute those intermediates regardless of input.

Figure 12 is a similar plot for the transformer $T_{4,6}^\infty$ with depth $L = 6$, evaluated on the task hop_{16}. The functionalities of the heads visualized in Figure 10 can be observed in the corresponding inner products. The collection of all inner products presents further evidence that the pointer-doubling phenomenon occurs in the trained models, due to the increase in compositions present in the largest inner products of deeper attention units.

While Figures 11 and 12 showcase the decisive alignment between self-attention heads and particular partial computations find^j in successfully trained models, Figure 13 demonstrates the loss of that decisiveness in poorly performing transformers. There, we visualize the alignments of the trained depth-4 transformer $T_{4,4}^\infty$ evaluated on hop_{16}, in which it attains a 61% token error. While a self-attention units in the second layer coincides with find^1, no strong correlations emerge deeper in the model. Unlike the other figures, the deeper self-attention units are "indecisive," lacking any large inner products and failing in particular to correlate with any highly compositional targets. This provides a visual explanation of the transformer's failure, since it lacked the effective representational capacity needed to learn a circuit with consistent and highly-compositional outputs.[9]

Alignment between Attention Heads and find^j for all hop_k Sub-Tasks (Figures 14 and 15). For an even more global lens on the mechanistic interpretability of these trained models, we visualize how the maximum inner products of each self-attention unit change for a fixed transformer for different sub-tasks hop_k. Figures 14 and 15 do so for the depth-4 and depth-6 networks respectively. The hue of each cell (and its numerical label) corresponds to the j^* with the most correlated inner product with corresponding attention unit $A^{\ell,h}$ in samples from $\mathrm{dom}(\mathrm{hop}_k)$, and the opacity corresponds to the magnitude of that inner product.

The takeaways of the previous inner product figures are apparent in these: the approximate doubling for the depth $L = 6$ transformer can be visualized by the vertically changing opaque colors. Conversely, a separation can be observed between the tasks where the depth $L = 4$ transformer performs well and has "decisive" self-attention units deeper in the network and those where it does not.

Moreover, the figures (especially Figure 15) demonstrate that several self-attention units have a consistent function among samples from the same task, while adapting in function to different hop_k tasks. This is most apparent in head $h = 4$ of layer $\ell = 6$, where the self-attention head functions as $\mathrm{find}^1, \mathrm{find}^3, \mathrm{find}^5$ or find^7 depending on the complexity of the task.

[9] Since these experiments are in the small alphabet size $|\Sigma| = 4$ regime, this task performs better than random guessing due to inferential capabilities that are are powered by the high embedding dimension and do not require implementing a pointer-chasing algorithm. We suspect that the "checkerboard" patterns are powered by this inference.

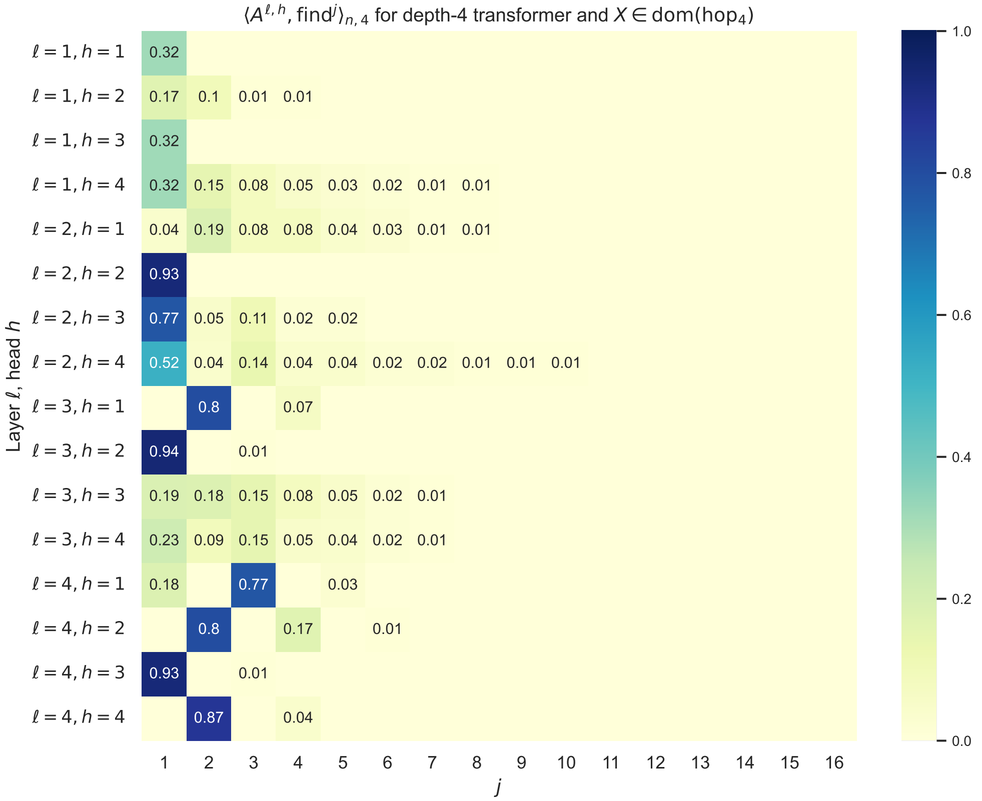
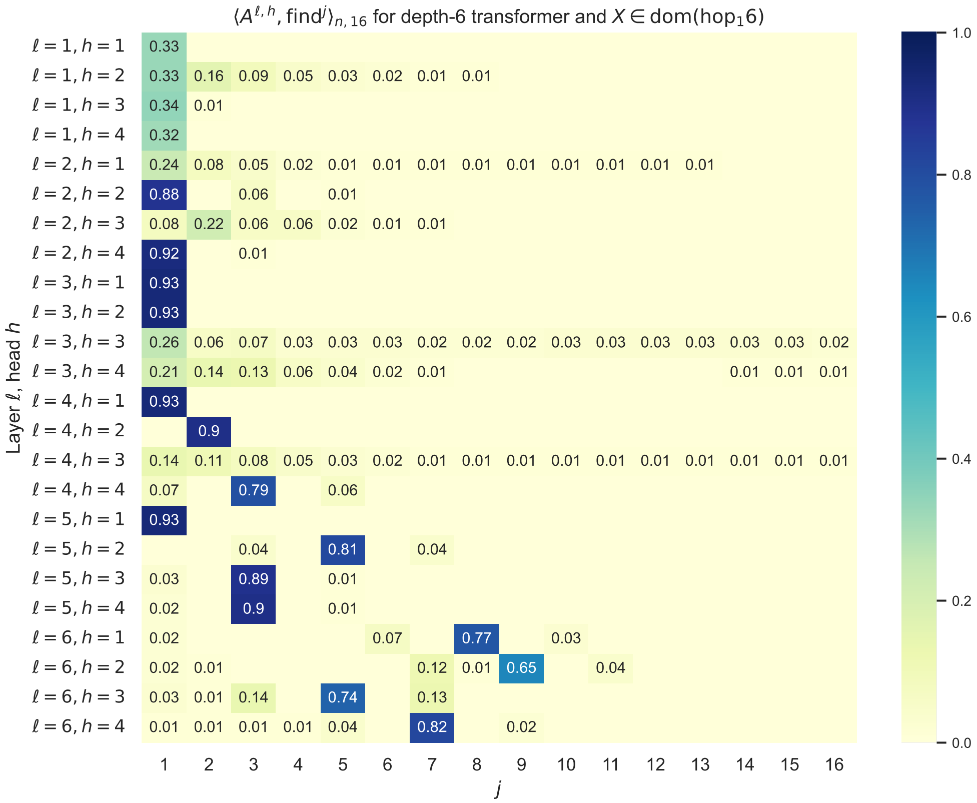


Figure 11. Plots of all inner products $\left\langle A^{\ell,h}[T_{4,4}^{\infty}], \mathrm{find}^j \right\rangle_{10,4}$ for $n = 10$ samples $X^1, \dots, X^{10} \in \mathrm{dom}(\mathrm{hop}_4)$ for the 4-layer transformer $T_{4,4}^{\infty}$.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$\ell = 1, h = 1$	0.33															
$\ell = 1, h = 2$	0.33	0.16	0.09	0.05	0.03	0.02	0.01	0.01								
$\ell = 1, h = 3$	0.34	0.01														
$\ell = 1, h = 4$	0.32															
$\ell = 2, h = 1$	0.24	0.08	0.05	0.02	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01			
$\ell = 2, h = 2$	0.88		0.06		0.01											
$\ell = 2, h = 3$	0.08	0.22	0.06	0.06	0.02	0.01	0.01									
$\ell = 2, h = 4$	0.92		0.01													
$\ell = 3, h = 1$	0.93															
$\ell = 3, h = 2$	0.93															
$\ell = 3, h = 3$	0.26	0.06	0.07	0.03	0.03	0.03	0.02	0.02	0.02	0.03	0.03	0.03	0.03	0.03	0.03	0.02
$\ell = 3, h = 4$	0.21	0.14	0.13	0.06	0.04	0.02	0.01							0.01	0.01	0.01
$\ell = 4, h = 1$	0.93															
$\ell = 4, h = 2$		0.9														
$\ell = 4, h = 3$	0.14	0.11	0.08	0.05	0.03	0.02	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
$\ell = 4, h = 4$	0.07		0.79		0.06											
$\ell = 5, h = 1$	0.93															
$\ell = 5, h = 2$			0.04		0.81		0.04									
$\ell = 5, h = 3$	0.03		0.89		0.01											
$\ell = 5, h = 4$	0.02		0.9		0.01											
$\ell = 6, h = 1$	0.02					0.07		0.77		0.03						
$\ell = 6, h = 2$	0.02	0.01					0.12	0.01	0.65		0.04					
$\ell = 6, h = 3$	0.03	0.01	0.14		0.74		0.13									
$\ell = 6, h = 4$	0.01	0.01	0.01	0.01	0.04		0.82		0.02							

Figure 12. Plots of all inner products $\left\langle A^{\ell,h}[T^{\infty}_{4,6}], \mathrm{find}^j \right\rangle_{10,16}$ for $n = 10$ samples $X^1, \dots, X^{10} \in \mathrm{dom}(\mathrm{hop}_{16})$ for the 6-layer transformer $T^{\infty}_{4,6}$.

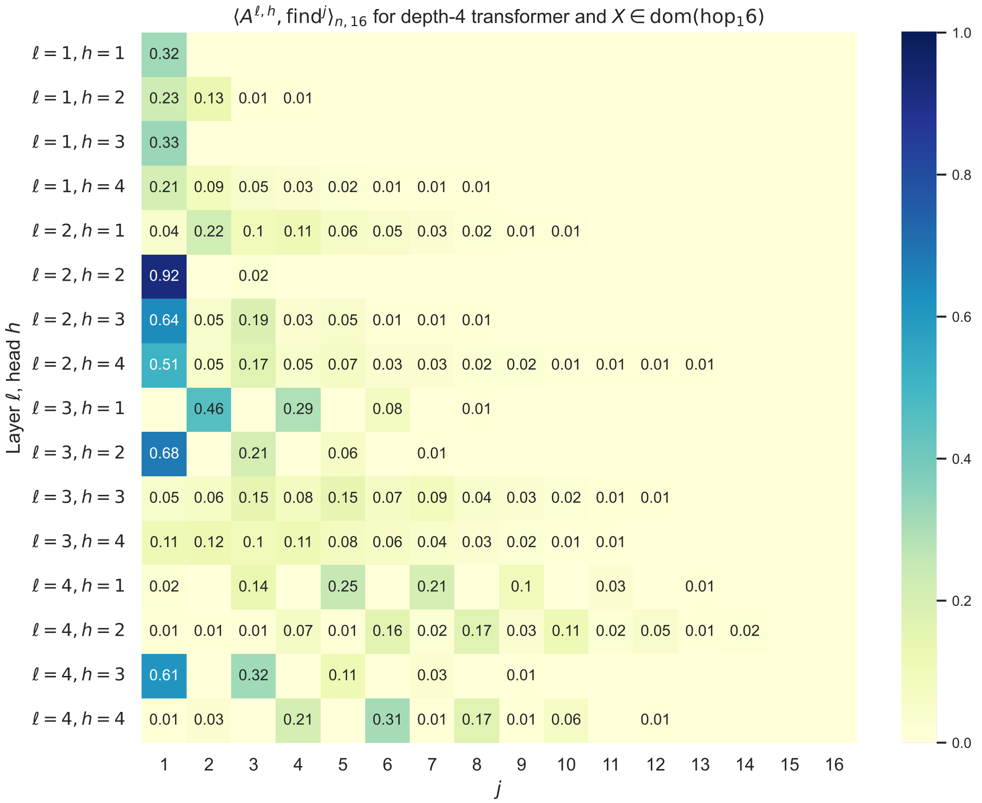


Figure 13. Plots of all inner products $\left\langle A^{\ell,h}[T^{\infty}_{4,4}], \mathrm{find}^j \right\rangle_{10,16}$ for $n = 10$ samples $X^1, \ldots, X^{10} \in \mathrm{dom}(\mathrm{hop}_{16})$ for the 4-layer transformer $T^{\infty}_{4,4}$.

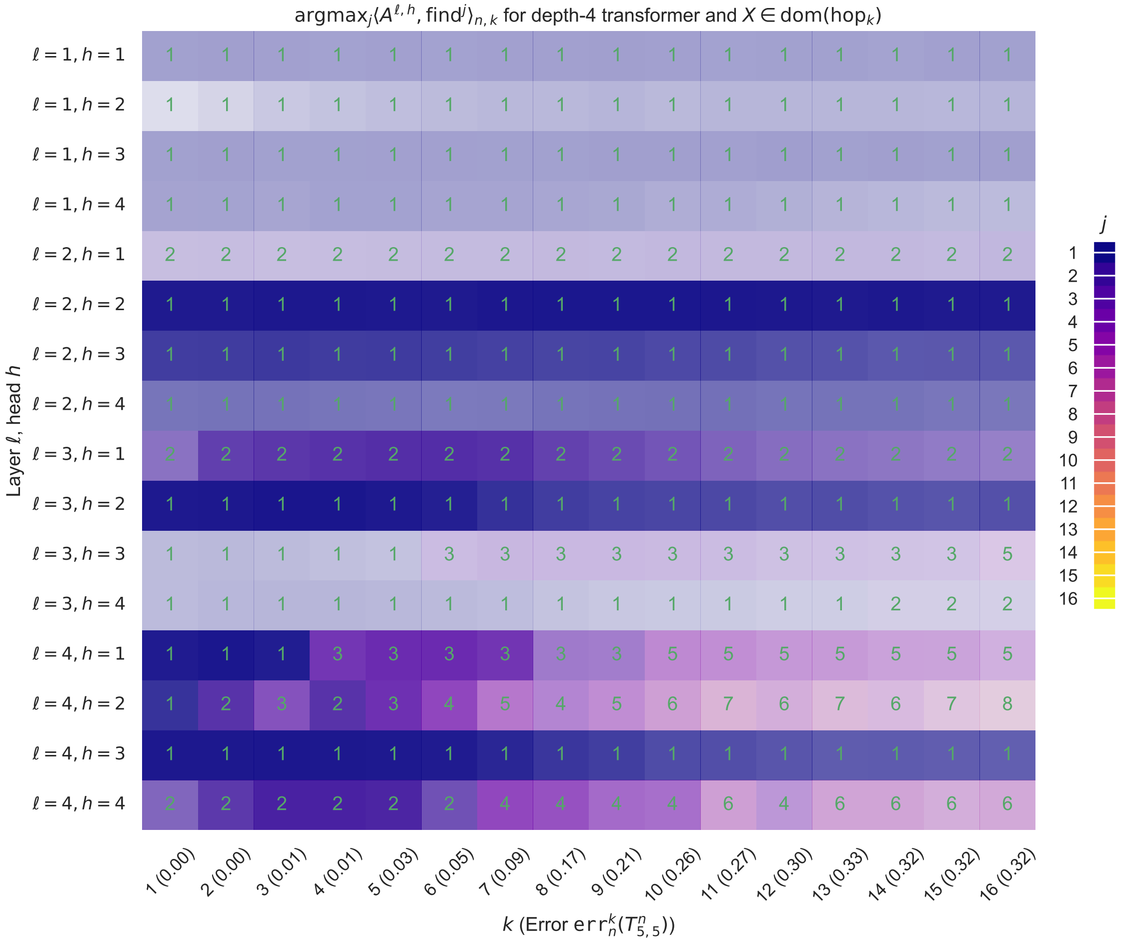


Figure 14. Plots of all the maximum inner products $\left\langle A^{\ell,h}[T_{4,4}^{\infty}], \mathrm{find}^j \right\rangle_{n,k}$ for $n = 10$ fixed samples $X^1, \dots, X^{10} \in \mathrm{dom}(\mathrm{hop}_k)$ for each $k \in [16]$ for the 4-layer transformer $T_{4,4}^{\infty}$. The hue corresponds to the index of the largest inner product $j^* = \arg\max_j \left\langle A^{\ell,h}[T_{4,4}^{\infty}], \mathrm{find}^j \right\rangle_{n,k}$, while the opacity is determined by the magnitude of the correlation.

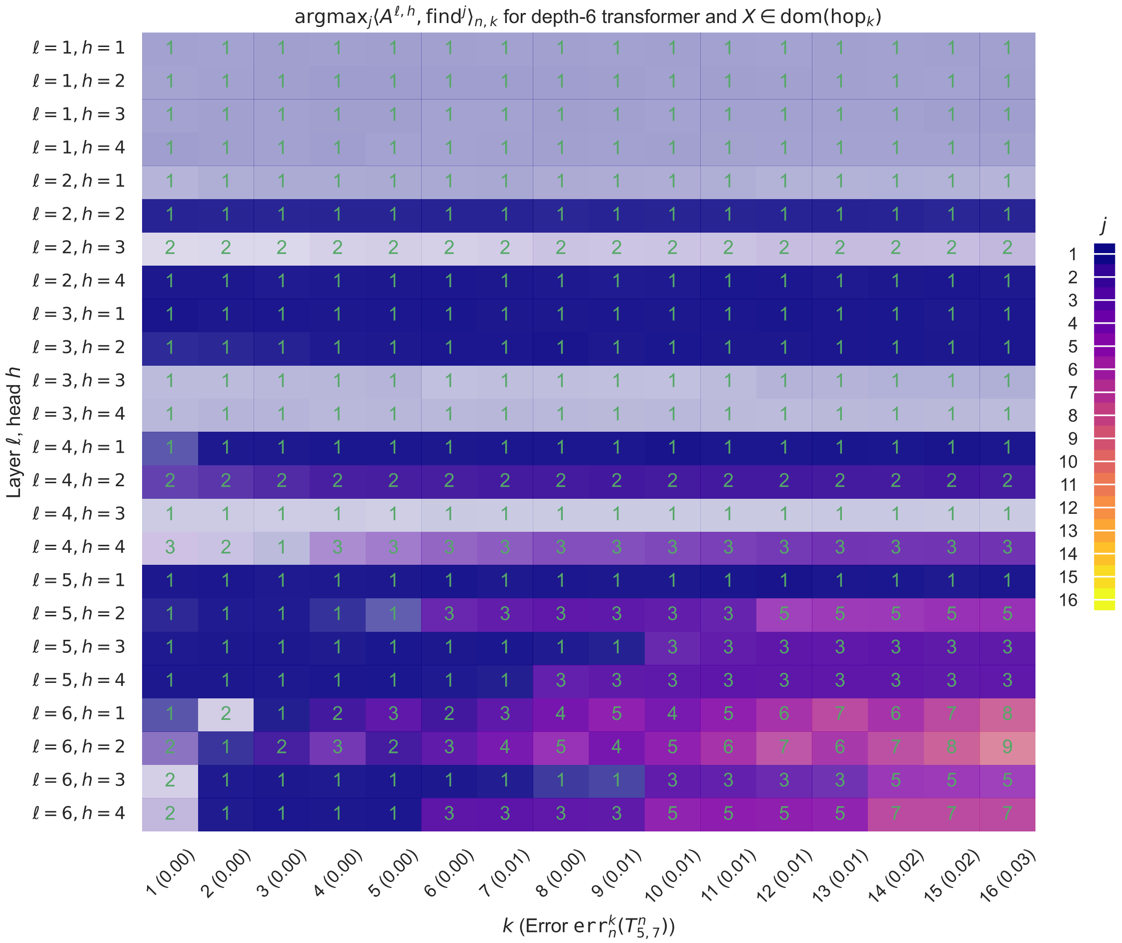


Figure 15. Plots of all the maximum inner products $\langle A^{\ell,h}[T_{4,6}^{\infty}], \text{find}^j \rangle_{n,k}$ for $n = 10$ fixed samples $X^1, \dots, X^{10} \in \text{dom}(\text{hop}_k)$ for each $k \in [16]$ for the 6-layer transformer $T_{4,6}^{\infty}$.

G.5 Finite-Sample Experiments (Empirical Claim G.3; Figures 16 to 19)

While most of our multi-hop experiments reside in the infinite-sample regime (where new samples are generated for every batch), we also trained several transformers on $n_{\text{train}} \in \{1000, 3000\}$ samples to evaluate whether generalization is possible in this domain, especially when the number of model parameters far exceeds the number of training samples. The two training set sizes expose a sharp threshold between two different generalization modes: low accuracy due to overfitting for most models on most tasks when $n_{\text{train}} = 1000$ and high accuracy approaching the infinite-sample regime when $n_{\text{train}} = 3000$.

Figure 16 compares the infinite-sample transformers $T_{4,L}^{\infty}$ with the 3000-sample models $T_{4,L}^{3000}$. 3000 training samples are sufficient to obtain comparable (if slightly worse) generalization error rates across model depths L and task complexities k. This supports a hypothesis that the existence of a small transformer that perfectly fits the data enables larger transformers to actually realize such architectures in the over-parameterized regime.

On the other hand, Figure 17 demonstrates that transformers trained on $n_{\text{train}} = 1000$ samples suffer poor performance on most tasks due to overfitting. While all models perform poorly on hop_k sub-tasks for large k, a depth-separation exists for simpler sub-tasks like hop_3. This suggests a positive inductive bias of deep transformers for simple compositional decision rules, which enables far better performance than other models in the overfitting regime.

To investigate this gap in performance, we contrast the self-attention inner products of depth-4 $T_{4,4}^{1000}$ and depth-6 $T_{4,6}^{1000}$ on the task hop_3 in Figures 18 and 19. The 6-layer model obtains a far superior classification error on the sub-task, and the interpretability plot establishes a plausible circuit it implements: It uses self-attention heads with find^1 functionality consecutively in layers 4, 5, and 6, which enables the robust retrieval of find^3 and hop_3. On the other hand, the 4-layer plot exhibits poor performance and only has two layers with find^1 functionality; this justifies the relatively strong performance of $T_{4,4}^{1000}$ on hop_2 and its poor performance on hop_3.

While neither model learns any kind of pointer-doubling construction, the 6-layer model is still able to learn a simple construction of hop_3 that the 4-layer model misses. The representational suitability of deeper models to compositional reasoning may thus provide a favorable inductive bias for learning the task in a setting with little data.

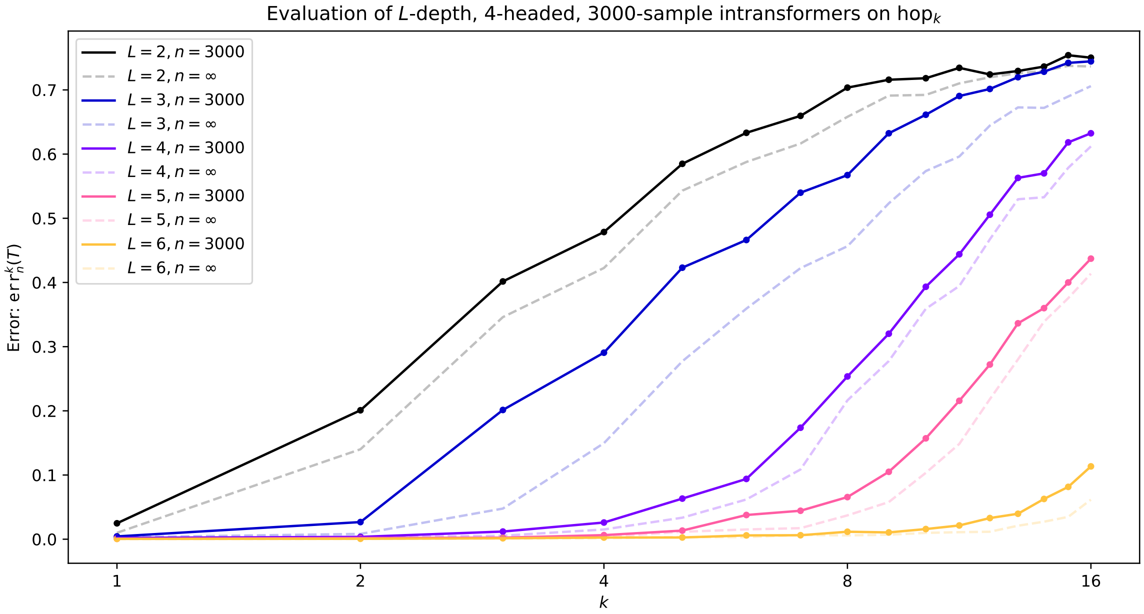


Figure 16. Comparison between the errors $\mathtt{err}_k^n(T_{4,L}^n)$ of transformers trained in the infinite sample regime (dashed line) and on $n_{\text{train}} = 3000$ samples (solid line) on the multi-hop task, evaluated on $n = 100$ samples per hop_k task.

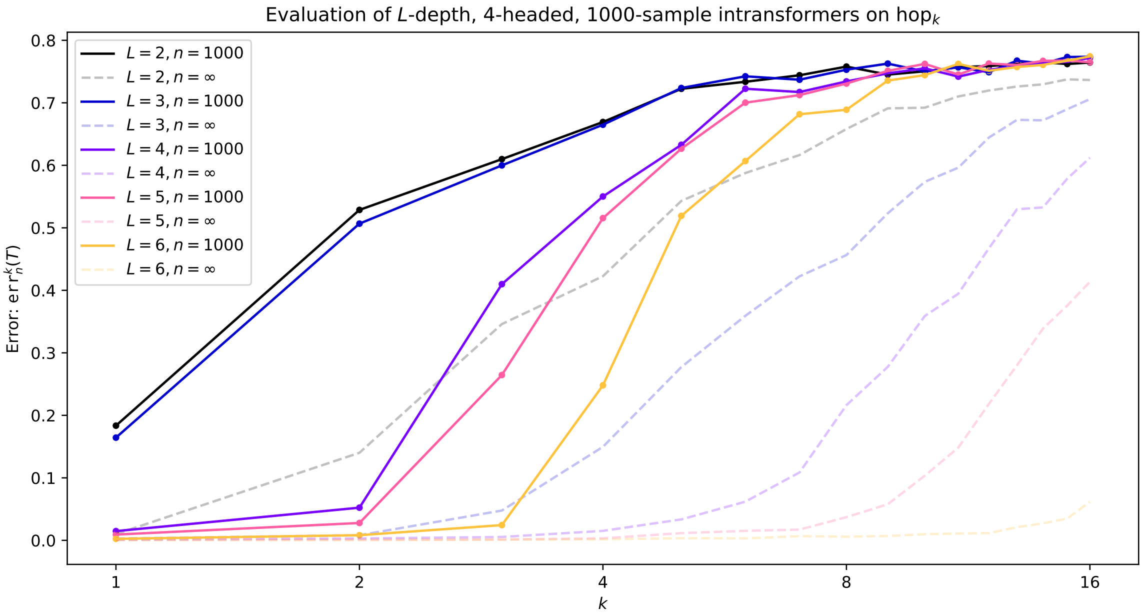
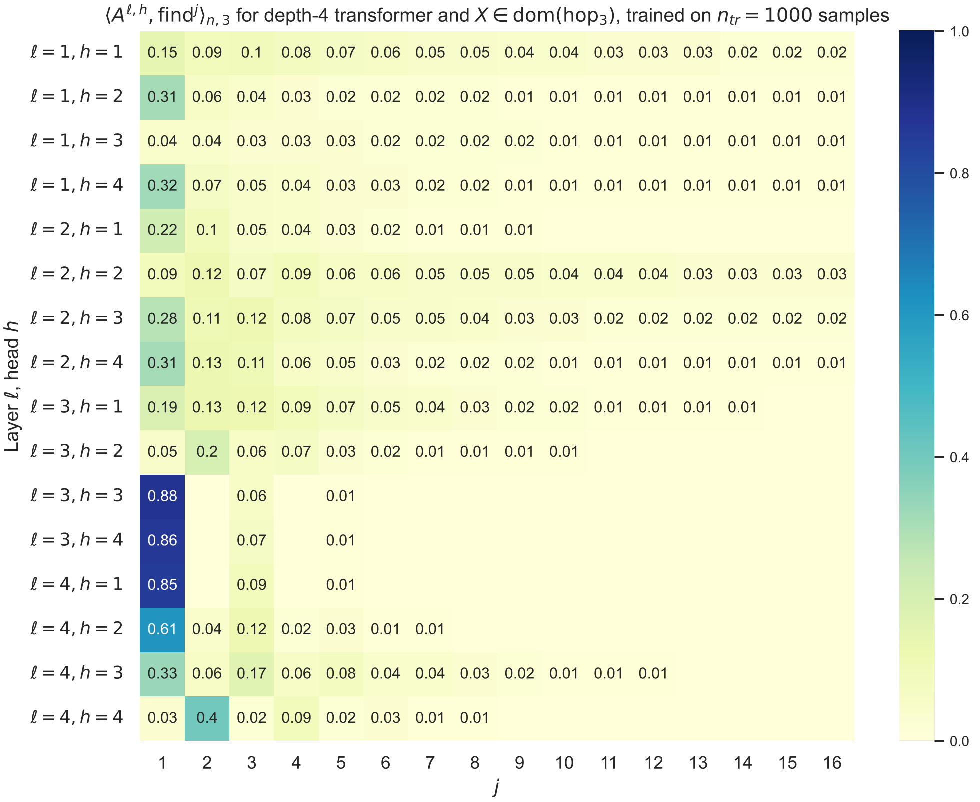


Figure 17. Comparison between the errors $\texttt{err}_k^n(T_{4,L}^n)$ of transformers trained in the infinite sample regime (dashed line) and on $n_{\text{train}} = 1000$ samples (solid line) on the multi-hop task, evaluated on $n = 100$ samples per hop_k task.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$\ell = 1, h = 1$	0.15	0.09	0.1	0.08	0.07	0.06	0.05	0.05	0.04	0.04	0.03	0.03	0.03	0.02	0.02	0.02
$\ell = 1, h = 2$	0.31	0.06	0.04	0.03	0.02	0.02	0.02	0.02	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
$\ell = 1, h = 3$	0.04	0.04	0.03	0.03	0.03	0.02	0.02	0.02	0.02	0.01	0.01	0.01	0.01	0.01	0.01	0.01
$\ell = 1, h = 4$	0.32	0.07	0.05	0.04	0.03	0.03	0.02	0.02	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
$\ell = 2, h = 1$	0.22	0.1	0.05	0.04	0.03	0.02	0.01	0.01	0.01							
$\ell = 2, h = 2$	0.09	0.12	0.07	0.09	0.06	0.06	0.05	0.05	0.05	0.04	0.04	0.04	0.03	0.03	0.03	0.03
$\ell = 2, h = 3$	0.28	0.11	0.12	0.08	0.07	0.05	0.05	0.04	0.03	0.03	0.02	0.02	0.02	0.02	0.02	0.02
$\ell = 2, h = 4$	0.31	0.13	0.11	0.06	0.05	0.03	0.02	0.02	0.02	0.01	0.01	0.01	0.01	0.01	0.01	0.01
$\ell = 3, h = 1$	0.19	0.13	0.12	0.09	0.07	0.05	0.04	0.03	0.02	0.02	0.01	0.01	0.01	0.01		
$\ell = 3, h = 2$	0.05	0.2	0.06	0.07	0.03	0.02	0.01	0.01	0.01	0.01						
$\ell = 3, h = 3$	0.88		0.06		0.01											
$\ell = 3, h = 4$	0.86		0.07		0.01											
$\ell = 4, h = 1$	0.85		0.09		0.01											
$\ell = 4, h = 2$	0.61	0.04	0.12	0.02	0.03	0.01	0.01									
$\ell = 4, h = 3$	0.33	0.06	0.17	0.06	0.08	0.04	0.04	0.03	0.02	0.01	0.01	0.01				
$\ell = 4, h = 4$	0.03	0.4	0.02	0.09	0.02	0.03	0.01	0.01								

Figure 18. Plots of all inner products $\langle A^{\ell,h}[T_{4,4}^{1000}], \text{find}^j \rangle_{10,3}$ for $n = 10$ samples $X^1, \ldots, X^{10} \in \text{dom}(\text{hop}_3)$ for the 4-layer transformer $T_{4,4}^{1000}$.

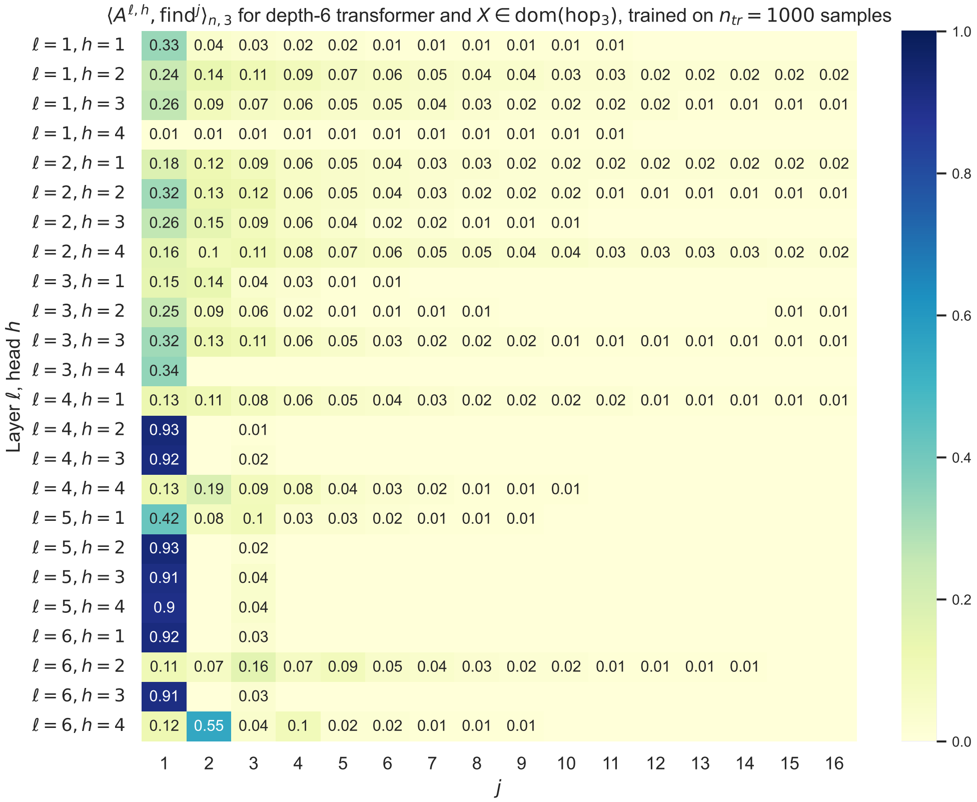


Figure 19. Plots of all inner products $\langle A^{\ell,h}[T_{4,6}^{1000}], \text{find}^j \rangle_{10,3}$ for $n = 10$ samples $X^1, \dots, X^{10} \in \text{dom}(\text{hop}_3)$ for the 6-layer transformer $T_{4,6}^{1000}$.