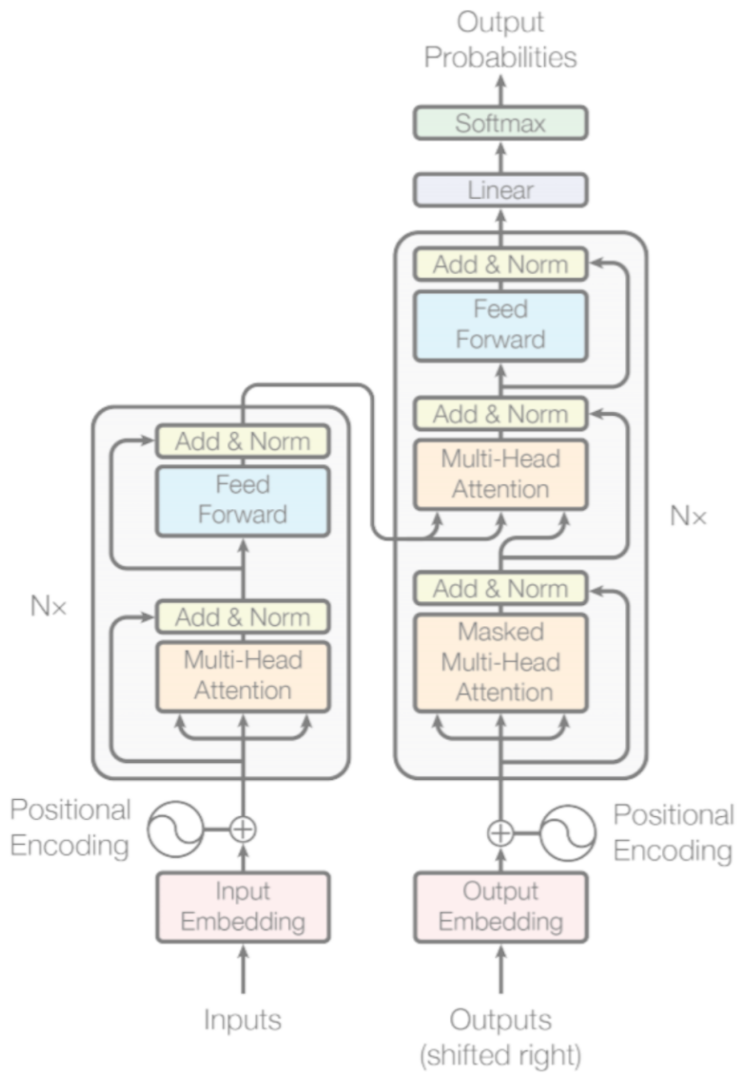
Viewing Log-Depth Transformers via the Lens of Distributed Computing

OptiML Group Meeting
October 10th, 2024



$\simeq$

Presented by Hanseul Cho

OptiML
Optimization & Machine Learning

Overview

- A new theoretical tool to understand the expressive power of transformers: **Massively Parallel Computation (MPC)**
- Depth separation: transformers > alternative architectures.
  - Based on a toy task: k**-hop induction task**

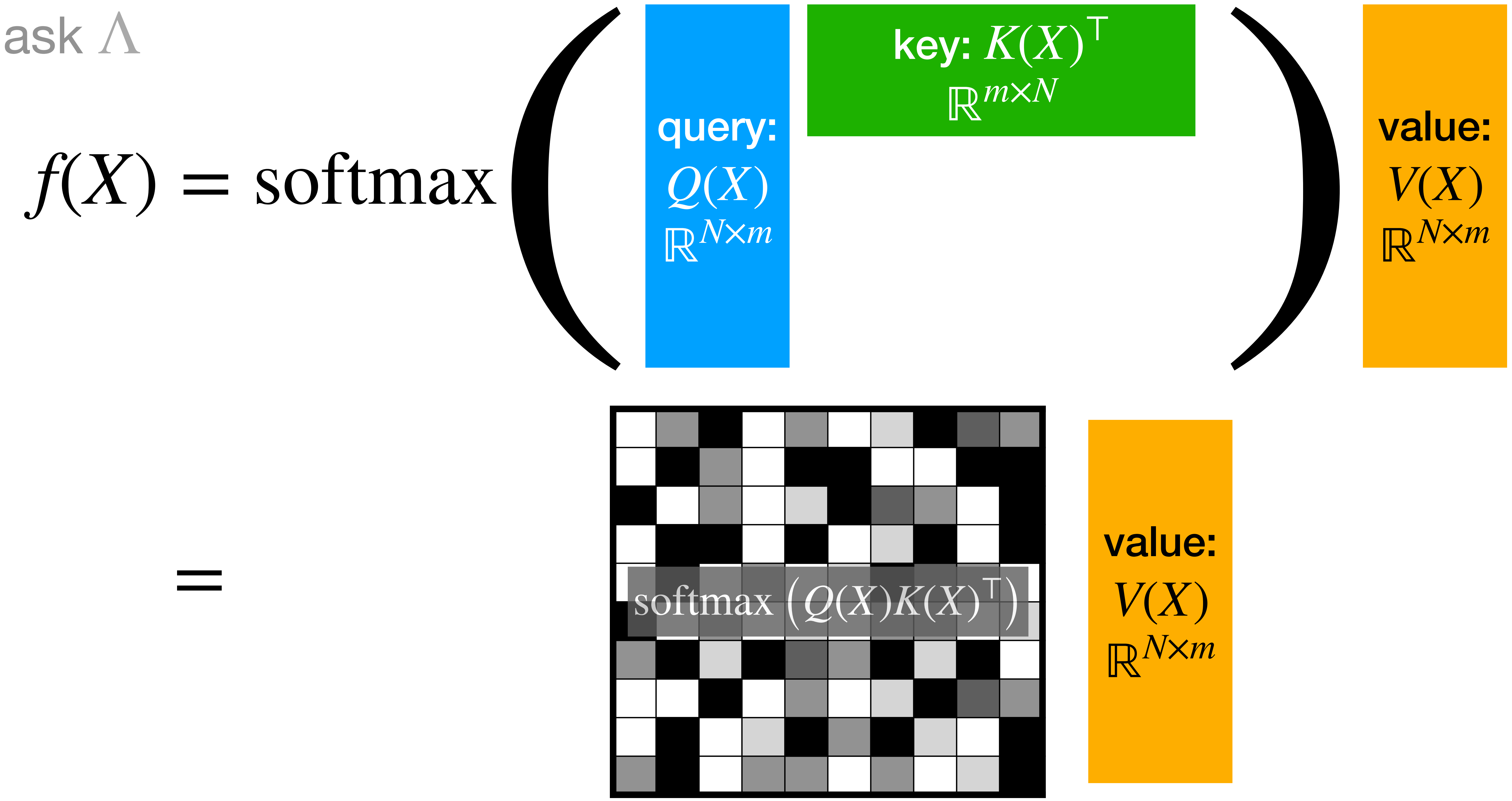
- Main references:
  - **[SHT24]** Clayton Sanford, Danial Hsu, and Matus Telgarsky. Transformers, Parallel Computation, and Logarithmic Depth. ICML 2024.
  - **[SFHT+24]** Clayton Sanford, Bahare Fatemi, Ethan Hall, Anton Tsitsulin, Mehran Kazemi, Jonathan Halcrow, Bryan Perozzi, and Vahab Mirrokni. Understanding Transformer Reasoning Capabilities via Graph Algorithms. arXiv preprint. 2024.

Contents

1. Transformer Architecture
2. Massively Parallel Computation (MPC) Model
3. Transformer ↔️ MPC Protocol
4. Separation between Architectures with k-hop Induction Head Task

Transformer Architecture

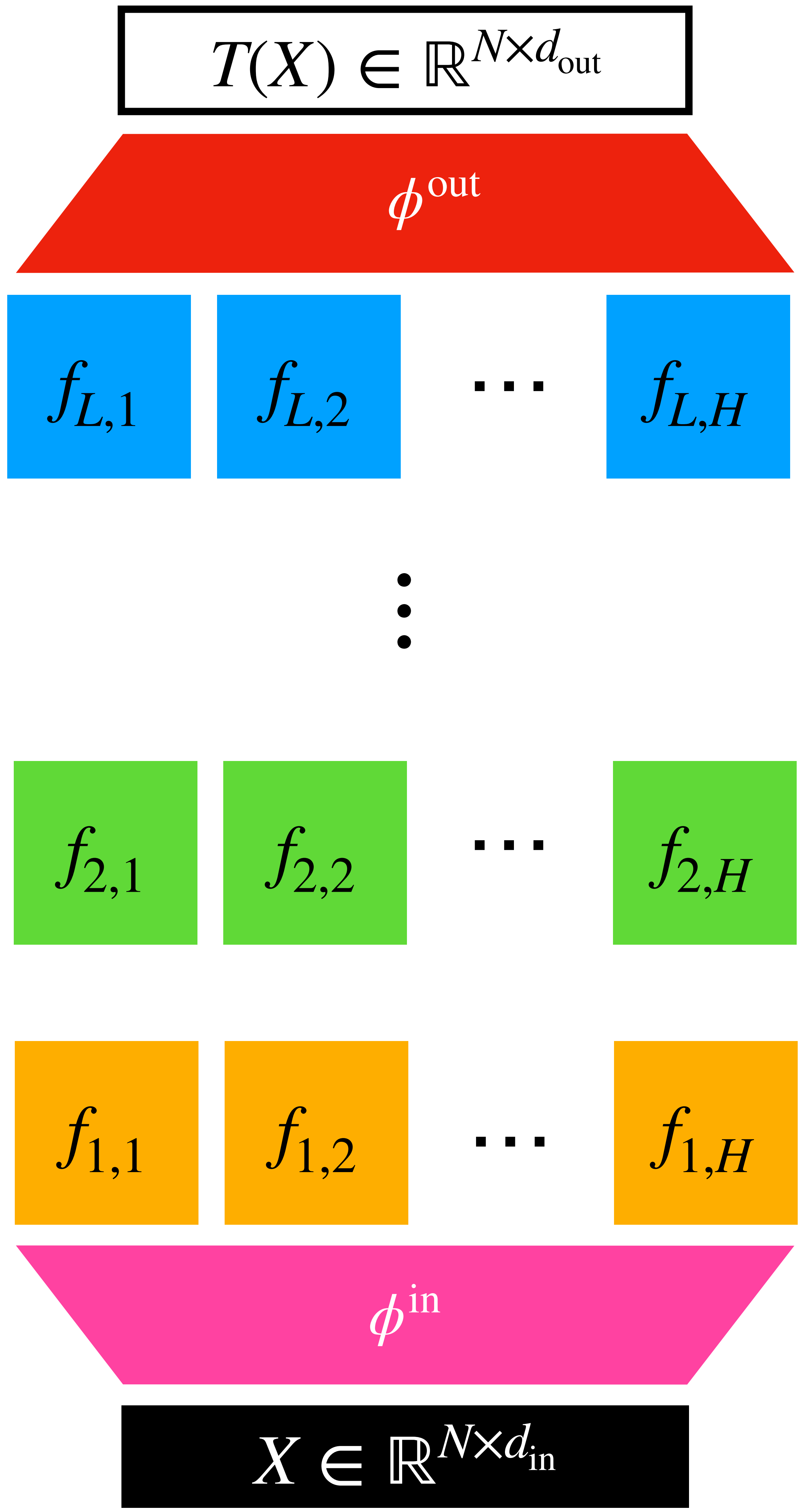
- **Self-attention head:**
  - $f(X) = \mathrm{softmax}\left(Q(X)K(X)^\top + \Lambda\right) V(X)$
  - $f, Q, K, V : \mathbb{R}^{N\times m} \to \mathbb{R}^{N\times m}$, row-wise softmax
  - $X = [x_1, \cdots, x_N]^\top$, attention mask Λ



* No positional embedding
* No Normalization layers
* $p = \Theta(\log N)$-bit precision

Transformer Architecture

- **Self-attention head:**
  - $f(X) = \mathrm{softmax}\left(Q(X)K(X)^\top + \Lambda\right) V(X)$
  - $f, Q, K, V : \mathbb{R}^{N\times m} \to \mathbb{R}^{N\times m}$, row-wise softmax
  - $X = [x_1, \cdots, x_N]^\top$, attention mask $\Lambda \in \{-\infty, 0\}^{N\times N}$
- **Multi-head attention (with residual connection):**
  - $g_l(X) = X + \sum_{h=1}^{H} f_{l,h}(X)$
- **Multi-layer perceptrons (MLPs) per word (or row):**
  - $\phi^{\mathrm{in}} : \mathbb{R}^{N\times d_{\mathrm{in}}} \to \mathbb{R}^{N\times m}$, $\phi^{\mathrm{out}} : \mathbb{R}^{N\times m} \to \mathbb{R}^{N\times d_{\mathrm{out}}}$
- **Transformer $T \in \mathrm{Transformer}^N_{m,L,H,d_{\mathrm{in}},d_{\mathrm{out}}}$:**
  - $T(X) = (\phi^{\mathrm{out}} \circ g_L \circ \cdots \circ g_1 \circ \phi^{\mathrm{in}})(X)$



Theoretical Lenses to Study Transformers

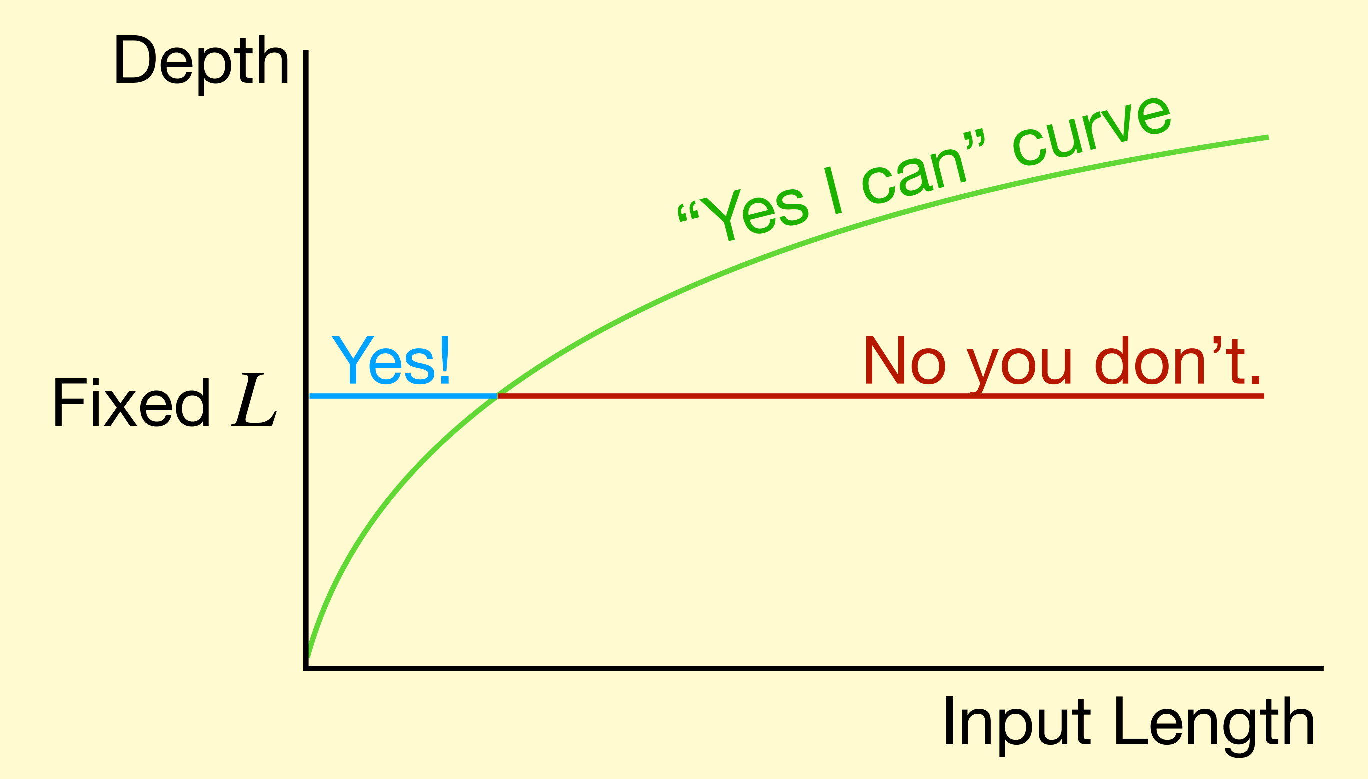
Specifically about their Expressive Power

1. Transformers as *machines that recognize formal languages*
   - Dyck language, star-free regular language …
   - **Fixed-size transformers** cannot handle long inputs
2. Transformers as *circuits*
   - TC^0, NC^1,… (do you remember?)
   - **Fixed-size transformers** cannot solve several *graph tasks* (e.g. graph connectivity)
3. Transformers as *finite-state automata*
   - Studies log-depth transformers but not even near-optimal

Theoretical Lenses to Study Transformers

Specifically about their Expressive Power

A plot in our mind: (not rigorous)



1. Transformers as *machines that recognize formal languages*
   - Dyck language, star-free regular language …
   - **Fixed-size transformers** cannot handle long inputs
2. Transformers as *circuits*
   - TC^0, NC^1,… (do you remember?)
   - **Fixed-size transformers** cannot solve several *graph tasks* (e.g. graph connectivity)
3. Transformers as *finite-state automata*
   - Studies log-depth transformers but not even near-optimal
4. **(New!) Transformers as *communication protocols* (e.g. MPC)**
   - Enable analysis via communication complexity & distributed computation
   - **Logarithmic-depth transformers** can solve several graph tasks (and they *might* be near-optimal!)

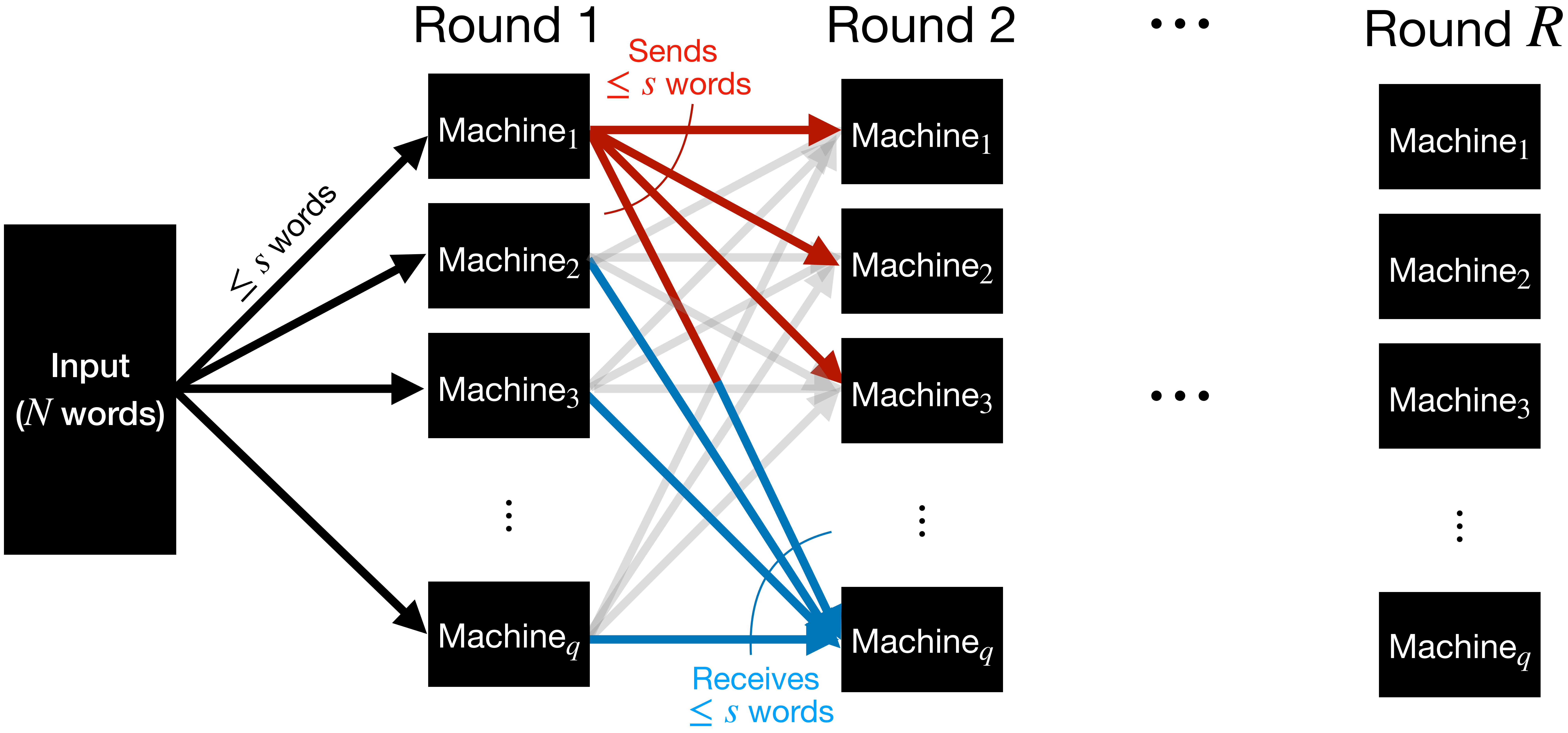
Massively Parallel Computation (MPC) [KSV10]

- A theoretical model to study ***distributed computing frameworks*** e.g. MapReduce, Hadoop, and Spark.
- Goal:
  - To design protocols that use fewer (e.g. $O(\log N)$) rounds of communication between machines whose local memory is sublinear in input length N.

Massively Parallel Computation (MPC) [KSV10]

- A theoretical model to study ***distributed computing frameworks*** e.g. MapReduce, Hadoop, and Spark.
- Goal:
  - To design protocols that use fewer (e.g. $O(\log N)$) rounds of communication between machines whose local memory is sublinear in input length N.
- Setup:
  - q machines with memory $s = o(N)$ (words). Total memory $qs = \Omega(N)$.
  - Computation proceeds in rounds:
  - In each round $r \in [R]$, each machine computes an arbitrary function of its local memory to prepare at most s words (in total) to send/distribute to other machines.

Massively Parallel Computation (MPC) [KSV10]



Massively Parallel Computation (MPC) [KSV10]

Example: Counting unique words (Recall from Data Mining 101…)

- Suppose we have an input with N, some of them are identical.
- Naive $O(N)$-time sequential algorithm: count one by one (use hash table).

Massively Parallel Computation (MPC) [KSV10]

Example: Counting unique words (Recall from Data Mining 101…)

- Suppose we have an input with N, some of them are identical.
- Naive $O(N)$-time sequential algorithm: count one by one (use hash table).
- **Two rounds** of MPC is enough! ($O(s)$-time in parallel)
  - Round 1:
    - Each machine i computes a word frequency count $c_{w,i}$ for each word w it has.
    - Choose a unique machine for each word (e.g., by hashing) and send the counts.
  - Round 2:
    - Add all received counts: $\sum_i c_{w,i}$. — Done.

Graph Connectivity Problem with MPC

“one-versus-two cycle” problem

- **Problem:** An undirected graph G with N vertices and N edges is given. Can you distinguish:
  - A single cycle on N vertices, and
  - A union of two cycles each on $N/2$ vertices?
- There exists a $O(\log N)$-round MPC protocol (See **here**, Section 7.2, Algorithm 13)
  - By serializing $G = (V, E)$ as $(u_1, v_1, u_2, v_2, \ldots, u_{|E|}, v_{|E|})$, where $E = \{(u_i, v_i)\}_{i=1}^{|E|}$.
- Can we do better?

Graph Connectivity Problem with MPC

“one-versus-two cycle” problem

- **Problem:** An undirected graph G with N vertices and N edges is given. Can you distinguish:
  - A single cycle on N vertices, and
  - A union of two cycles each on $N/2$ vertices?
- There exists a $O(\log N)$-round MPC protocol (See **here**, Section 7.2, Algorithm 13)
  - By serializing $G = (V, E)$ as $(u_1, v_1, u_2, v_2, \ldots, u_{|E|}, v_{|E|})$, where $E = \{(u_i, v_i)\}_{i=1}^{|E|}$.
- Can we do better? — A long-standing open problem! (But generally believed as “NO”.)
- **Conjecture (“one-versus cycle” conjecture).** Let any $\gamma > 0$, $\delta \in (0,1)$, and N. Then, any (γ, δ) -MPC protocol that solves the one-versus-two cycle problem *requires* $\Omega(\log N)$ rounds.
  - (γ, δ)-MPC protocol uses $q = \Theta(N^{1+\gamma-\delta})$ machines with memory of $s = O(n^{\delta})$ words.

MPC Protocol ➡️ Transformer

A R-round MPC Protocol can be simulated by a depth $O(R)$ transformer

- **Theorem [SFHT+24, Theorem 8].** For constants $0 < \delta < \delta' < 1$ and $\gamma > 0$, any R-round (γ, δ)-MPC protocol on N inputs can be expressed as a transformer $T \in \text{Transformer}^N_{m,L,H,1,1}$ with depth

$$L = O\left(\frac{R(1+\gamma)^2}{\min\{(\delta' - \delta)^2, \delta^2\}}\right),$$

single head $H = 1$ and embedding dimension $m = O(N^{\delta'})$.

MPC Protocol ➡️ Transformer

A R-round MPC Protocol can be simulated by a depth $O(R)$ transformer

- **Theorem [SFHT+24, Theorem 8].** For constants $0 < \delta < \delta' < 1$ and $\gamma > 0$, any R-round (γ, δ)-MPC protocol on N inputs can be expressed as a transformer $T \in \text{Transformer}^N_{m,L,H,1,1}$ with depth

$$L = O\left(\frac{R(1+\gamma)^2}{\min\{(\delta' - \delta)^2, \delta^2\}}\right),$$

  single head $H = 1$ and embedding dimension $m = O(N^{\delta'})$.

- **Corollary (Informal, SHT24 Cor. 3.3).** There exists a $O(\log N)$-layer 1-head transformer that identifies the connected components of any input graph (thus solving one-versus-two cycle problem).

MPC Protocol ⬅️ Transformer

A depth L transformer can be simulated by a $O(L)$-round MPC Protocol

- **Theorem [SHT24, Theorem 3.4].** For constants $0 < \delta < \delta' < 1$ and $\gamma > 0$, any transformer $T \in \text{Transformer}^N_{m,L,H,1,1}$ with width $mH = O(N^\delta)$ can be computed via a R-round $(1 + \delta', \delta')$-MPC protocol with

$$R = O\left(\frac{L}{\delta' - \delta}\right)$$

using $q = O(N^2)$ machines, each with $s = O(N^{\delta'})$ local memory.

MPC Protocol ⬅️ Transformer

A depth L transformer can be simulated by a $O(L)$-round MPC Protocol

- **Theorem [SHT24, Theorem 3.4].** For constants $0 < \delta < \delta' < 1$ and $\gamma > 0$, any transformer $T \in \text{Transformer}^{N}_{m,L,H,1,1}$ with width $mH = O(N^{\delta})$ can be computed via a R-round $(1+\delta', \delta')$-MPC protocol with

$$R = O\left(\frac{L}{\delta' - \delta}\right)$$

  using $q = O(N^2)$ machines, each with $s = O(N^{\delta'})$ local memory.
- **Corollary (Informal, SHT24 Cor. 3.5).** Assume the "one-versus-two cycle" conjecture. Then, for any constant $\epsilon > 0$, any transformer $T \in \text{Transformer}^{N}_{m,L,H,1,1}$ that solves the graph connectivity *requires* either a width $mH = \Omega(N^{1-\epsilon})$ or a depth $L = \Omega(\log N)$. (Thus, log-depth is near-optimal for parameter-efficient transformers.)

What we have so far…

- Connection between transformers ↔️ MPC protocols
  - One simulates another.
  - They share (in)abilities.
- (Im)possibility of solving the graph connectivity task.
  - Logarithmic depth can solve it (while constant depth cannot)
  - It *might be* optimal!

What we have so far…

- Connection between transformers ↔️ MPC protocols
  - One simulates another.
  - They share (in)abilities.
- (Im)possibility of solving the graph connectivity task.
  - Logarithmic depth can solve it (while constant depth cannot)
  - It *might be* optimal!
- What else we can say?
  - **The superiority of transformers above other alternatives.**
  - **… through a toy task 🧸.**

A Toy Task: k-hop Induction Heads (hop_k)

To study the separation between transformers versus non-parallel architectures

- From now: Decoder-only (causal) Transformers & Next-token Predictions.

A Toy Task: k-hop Induction Heads (hop_k)

To study the separation between transformers versus non-parallel architectures

- From now: Decoder-only (causal) Transformers & Next-token Predictions.
- Induction heads task
    - Find *the token that follows the last previous occurrence of the final token* in the input sequence.
    - E.g.) Given "…abcdebbdab", the answer is 'd'.

A Toy Task: k-hop Induction Heads (hop_k)

To study the separation between transformers versus non-parallel architectures

- From now: Decoder-only (causal) Transformers & Next-token Predictions.
- Induction heads task
  - Find *the token that follows the last previous occurrence of the final token* in the input sequence.
  - E.g.) Given "…abcdebbdab", the answer is 'd'.
- A generalization: k-hop induction heads task
  - E.g.) Given "…abcdebbdab", the answer of hop_2 (i.e., $k = 2$) is 'e'.
  - Motivated by *multi-step reasoning tasks*: "[…] Alice is a doctor. […] Bob's mother is Alice. […] What is the profession of Bob's mother?"

Transformers for k-hop Induction Heads

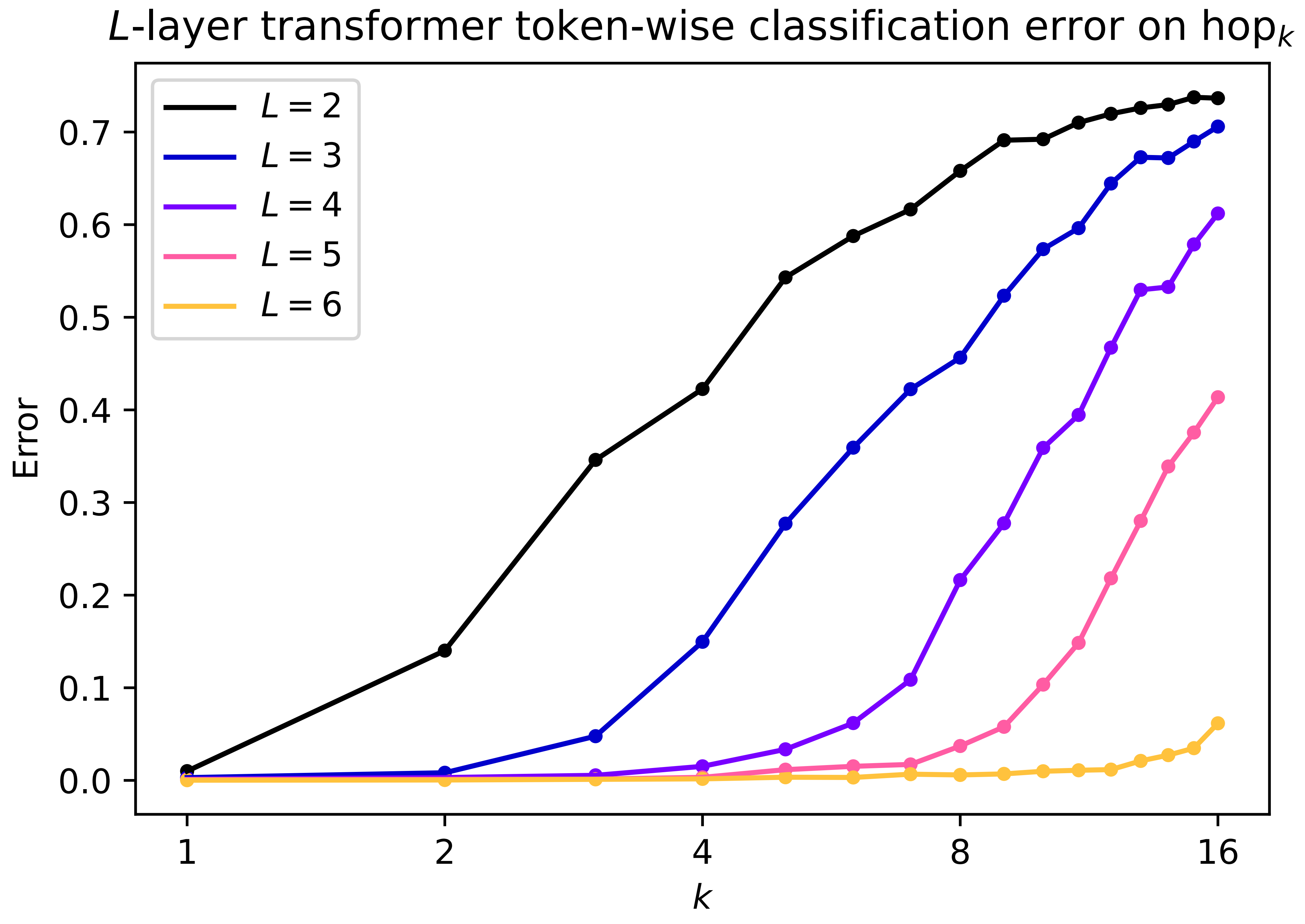
Depth $\Theta(\log k)$ is (maybe) necessary and (surely) sufficient

- For any $k \geq 1$ and a vocabulary of size $\leq N$:
- **Theorem (Sufficiency, SHT24 Thm 4.2).** There exists a $(\lfloor \log_2 k \rfloor + 2)$-layer 1-head transformer with causal attention masks and a constant embedding dimension that computes hop_k.
- **Theorem (Necessity, SHT24 Cor 4.3).** Assume the "one-versus-two cycle" conjecture. Consider any even $k = O(\sqrt{N})$. Then, any decoder-only transformer T that computes hop_k requires either a width $mH = \Omega(k^{0.99})$ or a depth $L = \Omega(\log k)$.
- The proof of the "necessity" is based on the proof of Theorem 3.4 of [SHT24] ("A transformer is simulated by an MPC protocol.") From the given T that computes hop_k, we construct a multi-round MPC protocol that converts one-or-two cycle graph into an input sequence X and then simulates $T(X)$ as its final output. In the end, $T(X)_N$ uniquely determines the number of cycles in the input graph.

Transformers for k-hop Induction Heads

Learned Transformers with Adam: Learnable with log-depth!

- **Empirical Setting:**
  - Curriculum learning mixture of hop_k for $k \in \{0, \ldots, 16\}$, vocab size 4.
  - Small GPT2-based models: $m = 128$, $H = 4$, $L \in \{2,3,4,5,6\}$, $N = 100$.
  - Training: 100K steps of Adam.
- **Observation:**
  - Sharp learnability threshold at $L \approx \lfloor \log_2 k \rfloor + 2$.



What about Other Architectures?

When $k = \Theta(N^{\xi})$ (for $\xi < 1/6$)

Architecture Type	Requirements ("Width OR Depth")	
	Width	Depth / N_{CoT}
Transformers with Dense Attention	$\Omega(N^{0.99\xi})$	$L = \Omega(\log N)$
Recurrent Architectures (RNN, Mamba, …)	$\Omega(N^{1-6\xi})$	$L = \Omega(N^{\xi})$
Transformers with Sub-Quadratic Attention	$\Omega(N^{1-6\xi})$	$L = \Omega(N^{\xi})$
1-Layer Transformers with Chain of Thoughts	$\Omega(N^{1-6\xi})$	$N_{\mathrm{CoT}} = \Omega(N^{\xi})$

* N_{CoT}: Additional number of tokens in the input sequence for CoT prompting

What about Other Architectures?

When $k = \Theta(\log N)$

Architecture Type	Requirements (“Width OR Depth”)	
	Width	Depth / N_{CoT}
Transformers with Dense Attention	$\Omega(\log^{0.99} N)$	$L = \Omega(\log \log N)$
Recurrent Architectures (RNN, Mamba, …)	$\tilde{\Omega}(N)$	$L = \Omega(\log N)$
Transformers with Sub-Quadratic Attention	$\tilde{\Omega}(N)$	$L = \Omega(\log N)$
1-Layer Transformers with Chain of Thoughts	$\tilde{\Omega}(N)$	$N_{\mathrm{CoT}} = \Omega(\log N)$

* N_{CoT}: Additional number of tokens in the input sequence for CoT prompting

Conclusion

- **Parallelism is a central feature of transformers.**
- Only a logarithmic scaling of depth and sublinear scaling of width (in N) suffices to build an expressive and well-performing transformer, even theoretically.
- (Near)-quadratic computation of attention seems necessary for log-depth transformers.
- Chain-of-thought (CoT) prompting is not enough for fixed-layer transformers to beat log-depth transformers without CoT.