Log-depth transformers and parallel computation

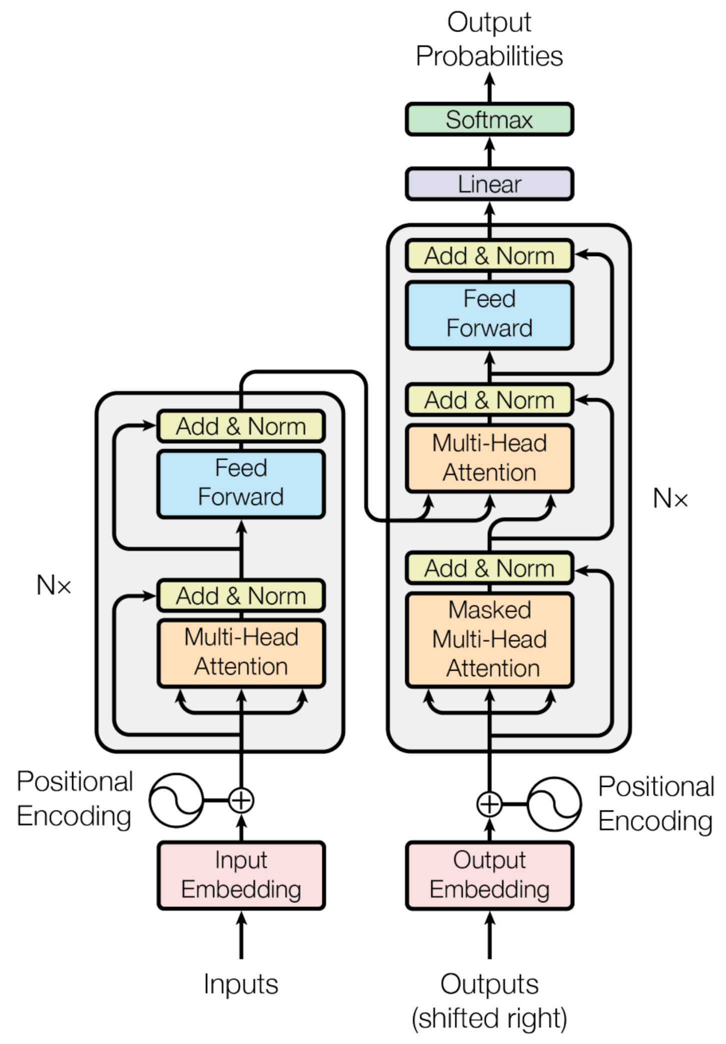
Clayton Sanford

February 19th, 2024

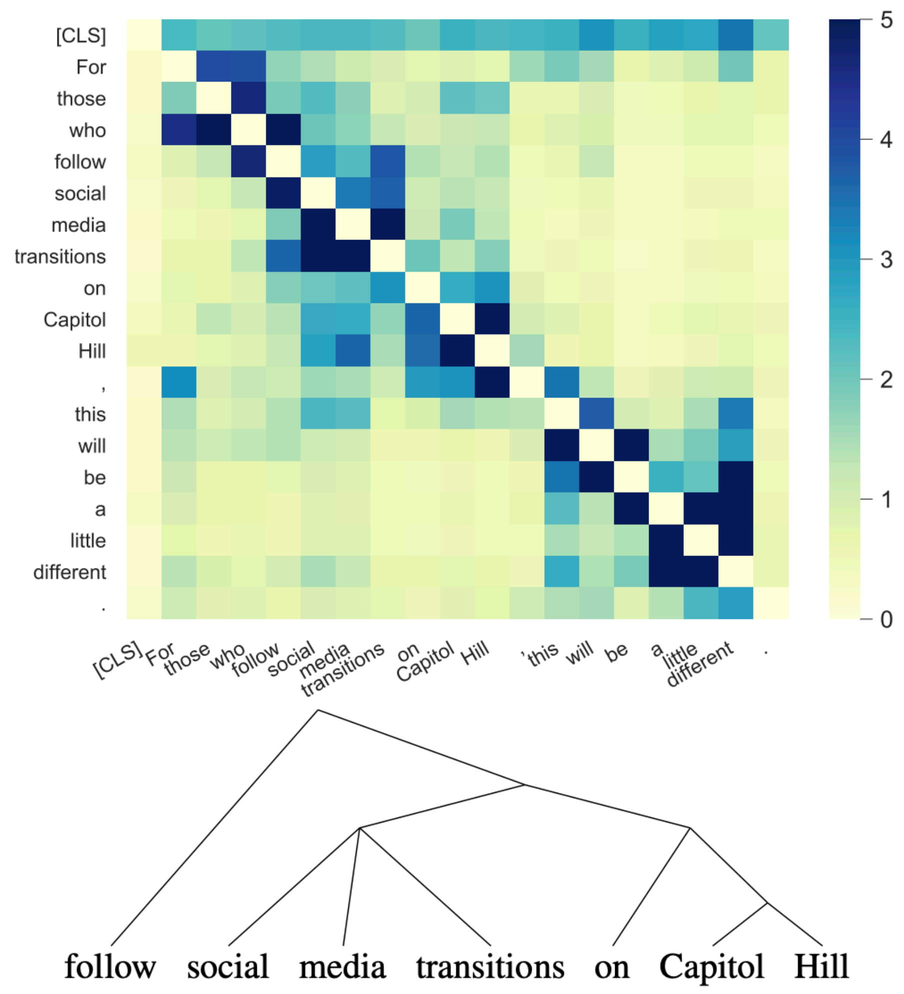
Joint work with Daniel Hsu and Matus Telgarsky

Transformer architecture

- **Sequence-to-sequence** architecture
- Backbone of modern large language models
- Replaced RNNs and LSTMs as state-of-the-art for NLP
- Characteristics:
  - Highly parallelizable
  - Core primitive: associative **self-attention** units
  - Scalable to long context length (32K GPT-4, 100K Claude)
  - Quadratic computational bottleneck



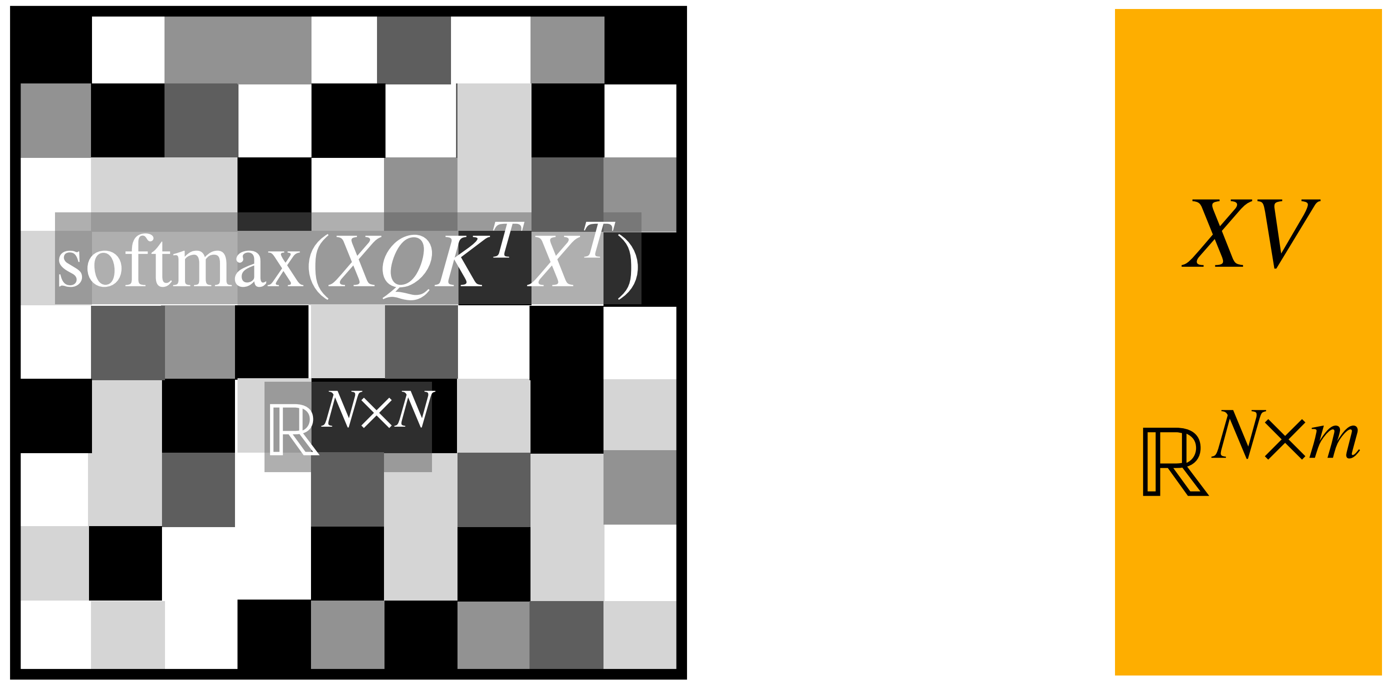
Vaswani et al ‘17



Rogers et al ’20

Transformer architecture

- **Self-attention unit:**
  $f(X) = \text{softmax}(XQK^TX^T)XV$
  for input $X \in \mathbb{R}^{N \times d}$,
  model parameters $Q, K, V \in \mathbb{R}^{d \times m}$.
- **Multi-headed attention:**
  $$g(X) = X + \sum_{h=1}^{H} f_h(X)$$
- **Element-wise multi-layer perceptron (MLP):** $\phi(X) = (\phi(x_1), \ldots, \phi(x_N))$
- **Full transformer:**
  $T(X) = (\phi_L \circ g_L \circ \ldots \circ g_1 \circ \phi_0)(X)$



Transformer architecture

What is it?

- **Self-attention unit:** $f(X) = \mathrm{softmax}(XQK^T X^T)XV$ for input $X \in \mathbb{R}^{N\times d}$, model parameters $Q, K, V \in \mathbb{R}^{d\times m}$.
- **Multi-headed attention:** $$g(X) = X + \sum_{h=1}^{H} f_h(X)$$
- **Element-wise multi-layer perceptron (MLP):** $\phi(X) = (\phi(x_1), \dots, \phi(x_N))$
- **Full transformer:** $T(X) = (\phi_L \circ g_L \circ \dots \circ g_1 \circ \phi_0)(X)$

Our questions

Can the strengths and limitations of transformers be understood via function approximation?

1. Comparisons to other models (RNNs, GNNs, $o(N^2)$ attention)
2. Representational impact of m, H, L
3. Difficult tasks for transformers

Theoretical lenses on transformer abilities

1. Transformers as *formal language recognizers*
2. Transformers as *automata*
3. Transformers as *circuits*
4. **Transformers as *communication protocols***
   - New perspective: view N inputs as agents that communicate in regimented manner; analysis via communication complexity and distributed computation.
   - Quantitative bounds w.r.t. width and depth; incorporates distinct transformer structure; comparisons with other models.

Massively Parallel Computation [KSV10]

(not Multi-Party Computation)

- MPC = theoretical model of MapReduce
- A distributed protocol is (γ, δ)-**MPC** on $O(\log N)$-bit input of size N if:
  - q total machines, each having local memory $s = \Theta(N^\delta)$.
    - Global memory $qs = O(N^{1+\gamma})$.
  - Input divided among $\Theta(N/s)$ machines.
  - Each of r rounds, machines perform perform computation on their inputs, send and receive messages simultaneously.
  - Unbounded computation, total size of messages sent and received per machine $\leq s$.
- Think: $\gamma, \delta = 0.01$

Massively Parallel Computation [KSV10]

Examples

- Dynamic programming, maximum matching, clustering, …
- Graph connectivity, diameter estimation, spanning forest in logarithmic rounds [**Andoni**, Song, **Stein**, Wang, **Zhong** ’18]

Our results

Theorem 1: Transformers simulate MPC protocols.

↓

Log-depth Transformers can solve connected components & pointer chasing

* PC construction is empirically learnable by log-depth transformers!

Theorem 2: MPC protocols simulate transformers.

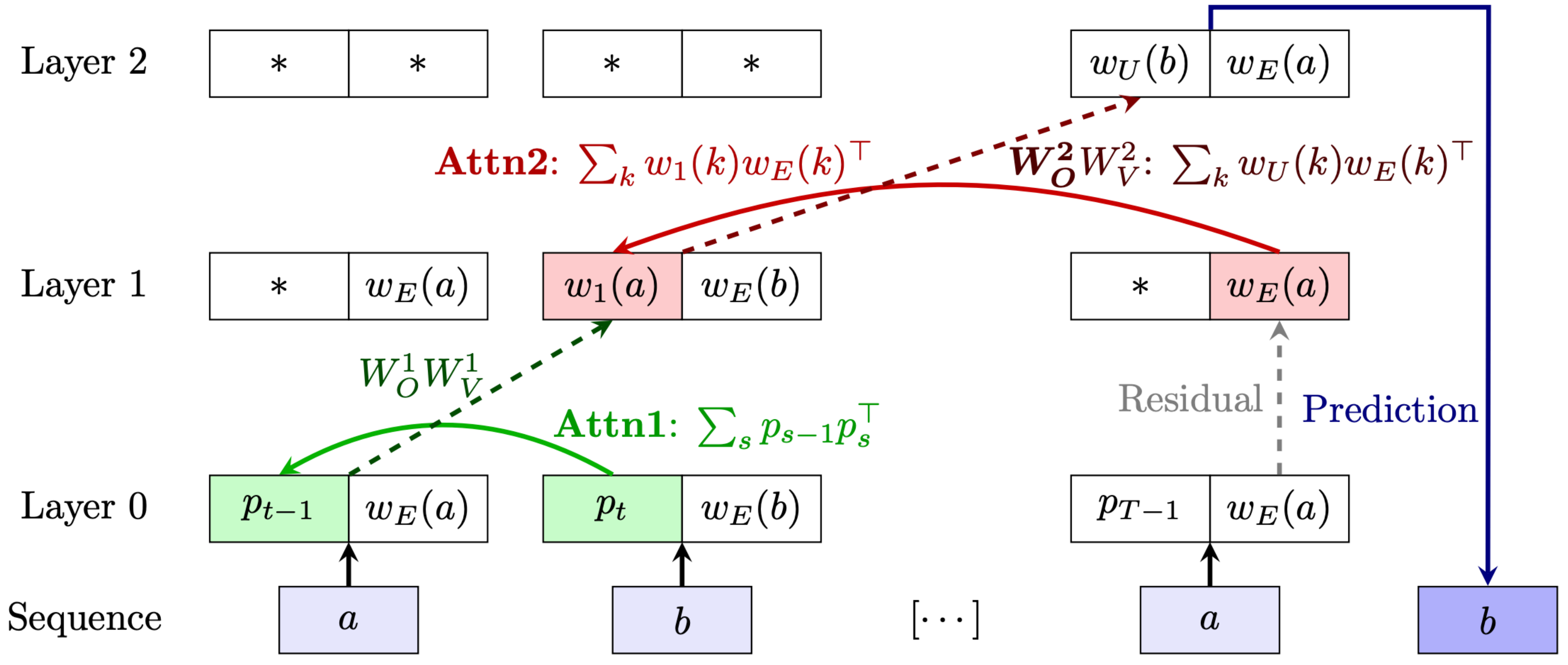
↓

Transformers require log-depth to solve CC & PC under MPC conjecture

* GNNs, RNNs, sub-quadratic attention models require poly-depth!

Induction heads: Powers of multi-layer transformers

- **Task**: Complete most recent matching bigram:
    - $\mathrm{IH}(\texttt{daccabcddca})_N = \texttt{b}$
- Occurs frequently as primitive in trained transformers [Anthropic]
- Natural construction with 2 layers [BCBJB23]:
    - Layer 1: identify previous token
    - Layer 2: find most recent occurrence of token



[BCBJB23]

Parallelism and Pointer Hopping

Idea: Capture powers of depth with recursive and parallelizable tasks.

Toy task: k-hop induction heads:

$$\mathrm{hop}_k(\ldots a_k a_{k+1} \ldots a_{k-1} a_k \ldots a_1 a_2 \ldots a_1) = a_{k+1}$$

MPC algorithms: Graph connectivity, minimum spanning tree, …

Results

1. $O(\log k)$-depth constructions.
2. $\Omega(\log k)$ depth lower bounds (conditional).
3. Separation from other architectures (GNN, multi-layer RNNs, some sub-quadratic-attention transformers).

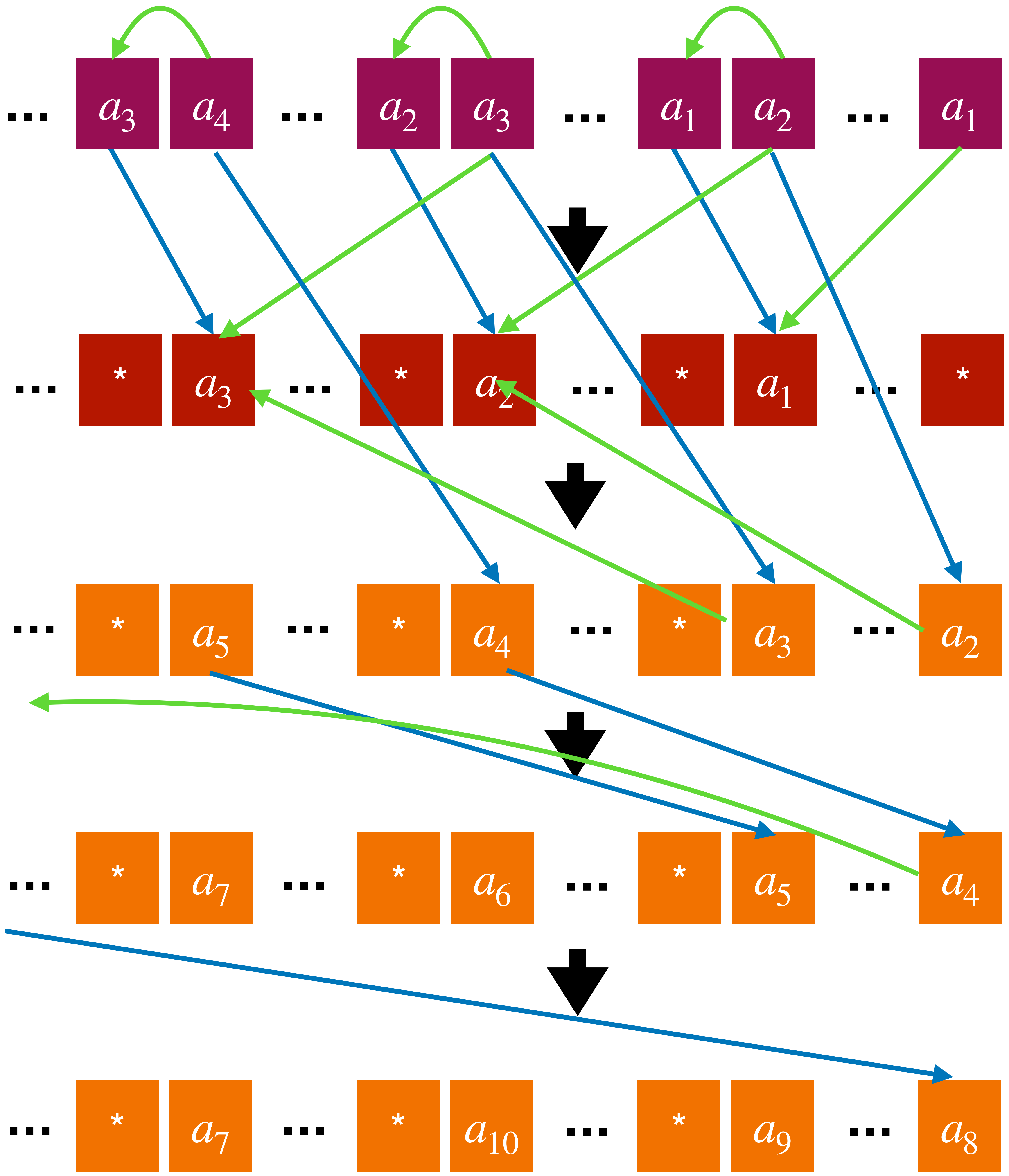
$\log(k)$-depth hop_k construction

Toy task: k-hop induction heads:

$$\text{hop}_k(\ldots a_k a_{k+1} \ldots a_{k-1} a_k \ldots a_1 a_2 \ldots a_1) = a_{k+1}$$

baebcabebdea.

Theorem: There exists a transformer with depth $L = O(\log k)$ and width $m = O(1)$ computing hop_k.



Learnability with gradient descent

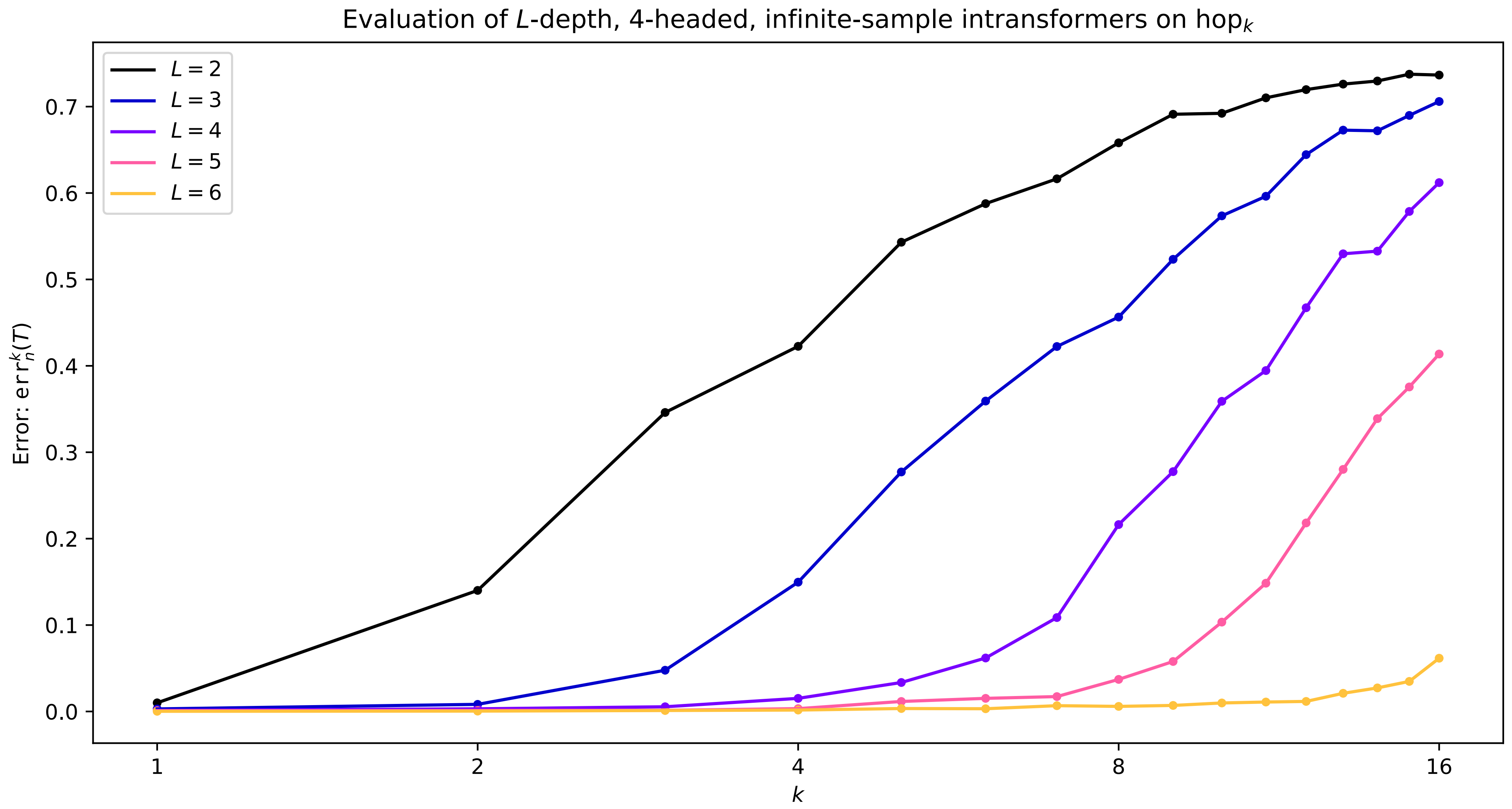
Toy task: k-hop induction heads:

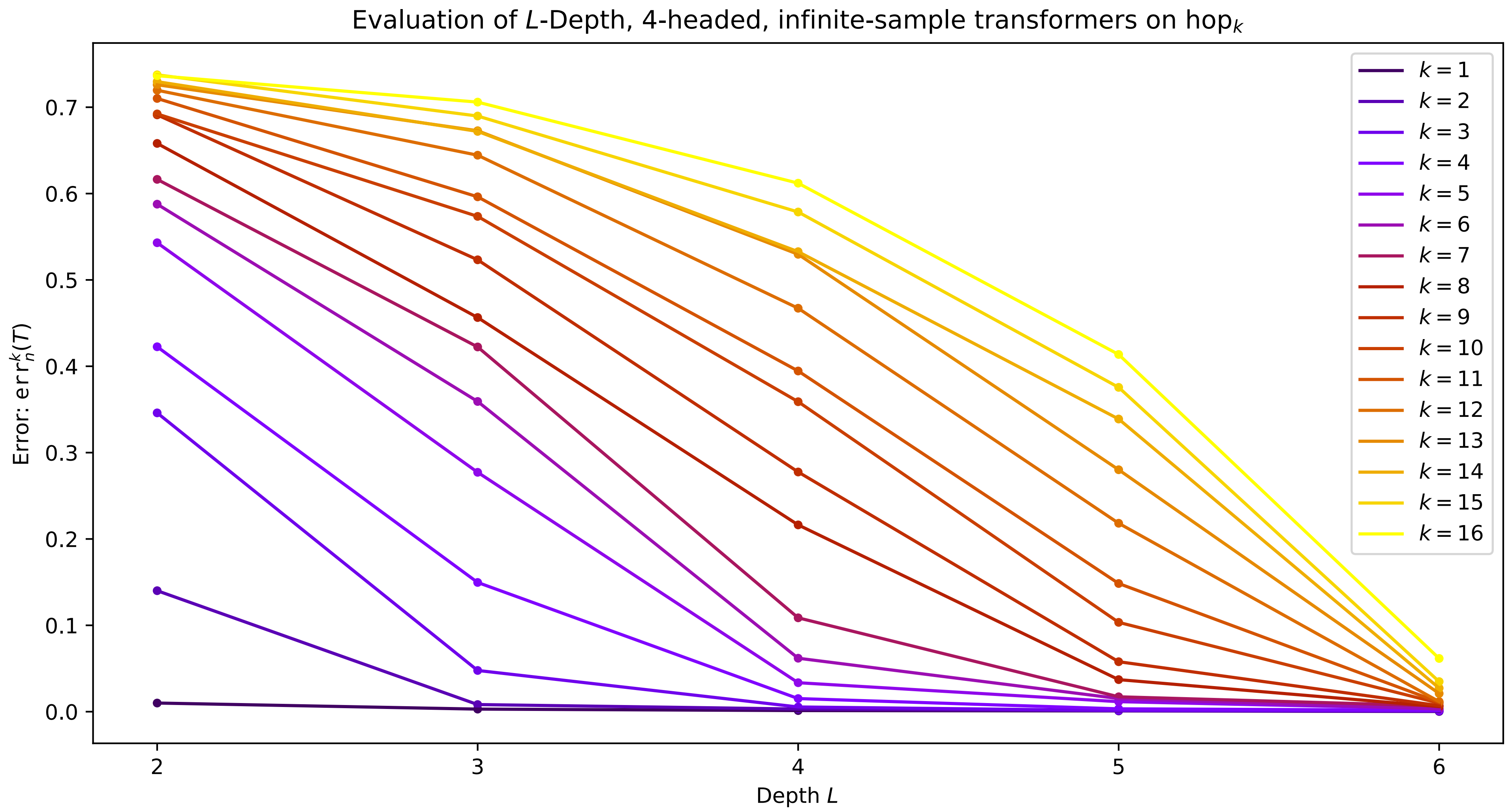
$$\text{hop}_k(\ldots a_k a_{k+1} \ldots a_{k-1} a_k \ldots a_1 a_2 \ldots a_1) = a_{k+1}$$

Empirical setting:

- Curriculum learning mixture of hop_k, $k \in \{0,1,\ldots,16\}$, 4 distinct tokens.
- Small GPT2 models: $m = 128$, $H = 4$, $L \in \{2,3,4,5,6\}$, $N = 100$.
- Training: 100K steps of Adam.

Sharp learnability threshold at $L = \log_2(k) + 2$





Learnability with gradient descent

Toy task: k-hop induction heads:

$$\mathrm{hop}_k(\ldots a_k a_{k+1} \ldots a_{k-1} a_k \ldots a_1 a_2 \ldots a_1) = a_{k+1}$$

Empirical setting:

- Curriculum learning mixture of hop_k, $k \in \{0,1,\ldots,16\}$, 4 distinct tokens.
- Small GPT2 models: $m = 128, H = 4,$ $L \in \{2,3,4,5,6\},$ $N = 100.$
- Training: 100K steps of Adam.

Self-attention head interpretability, $\mathrm{hop}_{16}, L = 6$

$\langle A^{\ell,h}, \mathrm{find}^j \rangle_{n,16}$ for depth-6 transformer and $X \in \mathrm{dom}(\mathrm{hop}_16)$

Layer ℓ, head h \ j	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							

Parallelism and Pointer Hopping

Further Questions

1. Which other tasks are solvable in $\log(k)$ depth?
2. Do more efficient constructions exist?

Reductions to and from **Massively Parallel Computation** (MPC) distributed computing protocol.

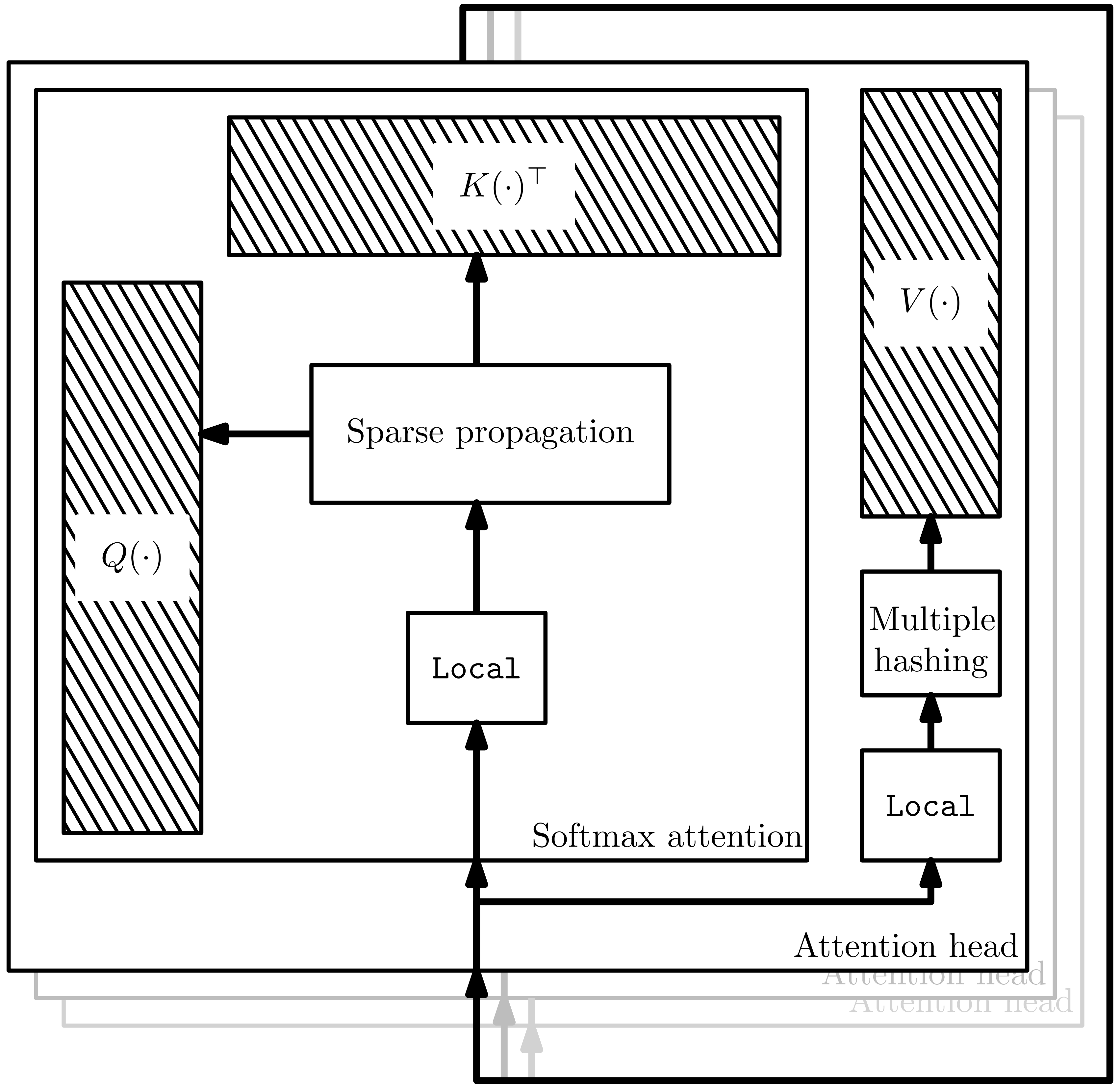
Relationships between transformers and MPC

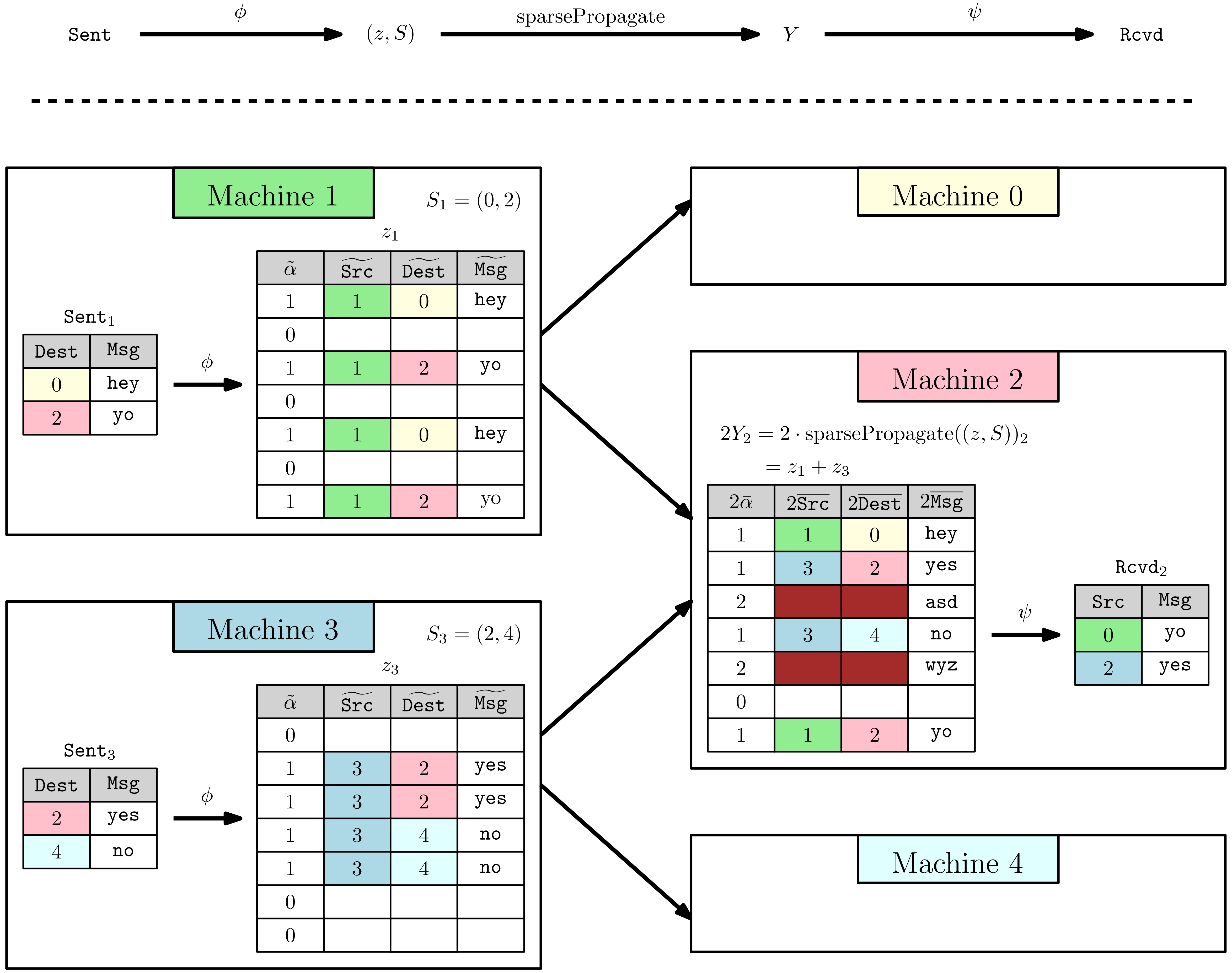
Theorem 1 [MPC to Transformers]: Any R-round (γ, δ)-MPC protocol can be simulated by a transformer with depth $L = O(R)$ and width $m = \tilde{O}(N^{4\delta})$.

- Main technical challenge: coordinating message passing.
  - Error correction via copies of messages in sparse locations on value vector XV.

Proof pictures…

Theorem 1 [MPC to Transformers]: Any R-round (γ, δ)-MPC protocol can be simulated by a transformer with depth $L = O(R)$ and width $m = \tilde{O}(N^{4\delta})$.





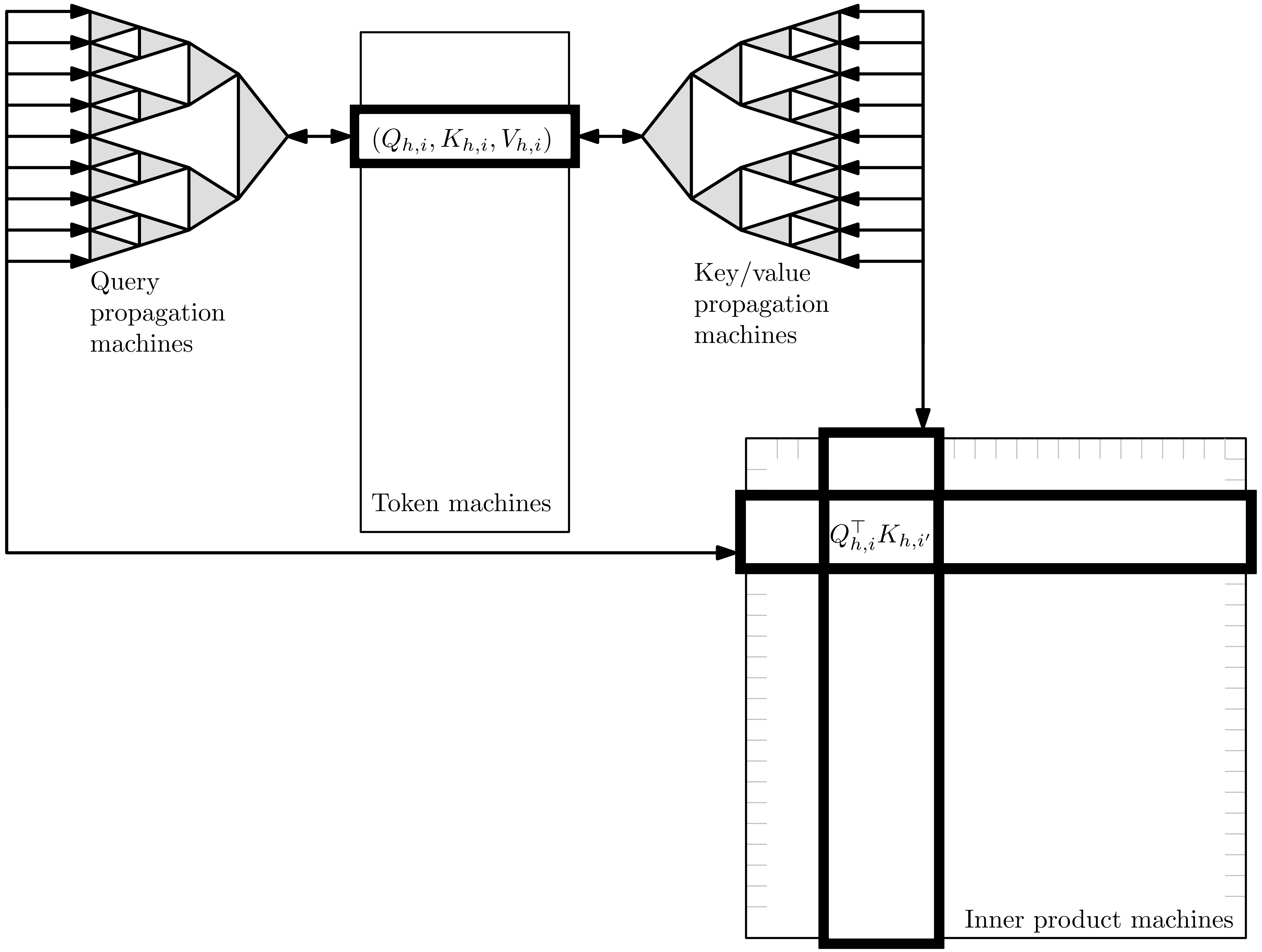
Relationships between transformers and MPC

Theorem 2 [Transformers to MPC]: Any transformer with depth L and width $m = O(N^{\delta})$ can be simulated by an $O(L/\delta)$-round $(1,2\delta)$-MPC protocol.

- Key limitation: quadratic scaling in global memory.
- Proof idea: Simulate each layer with “embedding machines” and “inner product machines”

Proof pictures…

Theorem 2 [Transformers to MPC]: Any transformer with depth L and width $m = O(N^{\delta})$ can be simulated by an $O(L/\delta)$-round $(1,2\delta)$-MPC protocol.



$\log(k)$-depth constructions for k-diameter graph problems

Theorem 1 [MPC to Transformers]: Any R-round (γ, δ)-MPC protocol can be simulated by a transformer with depth $L = O(R)$ and width $m = \tilde{O}(N^{4\delta})$.

Problem	MPC [ASSWZ18]	Transformer
Graph connectivity	(0.01, 0.01)-MPC, $R = O(\log N)$.	$L = O(\log N)$, $m = O(N^{0.04})$.
Min spanning forest	(0.01, 0.01)-MPC, $R = O(\log N)$.	$L = O(\log N)$, $m = O(N^{0.04})$.
hop_k	n/a	$L = O(\log k)$, $m = O(1)$.

Conjectured optimality of $\log(k)$ depth

Theorem 2 [Transformers to MPC]: Any transformer with depth L and width $m = O(N^{\delta})$ can be simulated by an $O(L/\delta)$-round $(1,2\delta)$-MPC protocol.

Conjecture: Every MPC with $\delta \in (0,1)$protocol distinguishing N-cycle graph from two $N/2$-cycles uses $r = \Omega(\log N)$ rounds.

➡ Transformers computing hop_k require $m = \Omega(k^{0.49})$ or $L = \Omega(\log k)$.

➡Transformers computing diameter **graph connectivity** require $mH = \Omega(N^{0.49})$ or $L = \Omega(\log N)$.

What about other architectures?

Graph Neural Networks (GNNs)

- [Loukas19] relates GNNs to CONGEST distributed computing model.
- GNNs require $L\sqrt{m} = \tilde{\Omega}(\sqrt{N})$ to evaluate subgraph connectivity.
  - vs $L = \log(N), m = N^{0.01}$ for transformers.

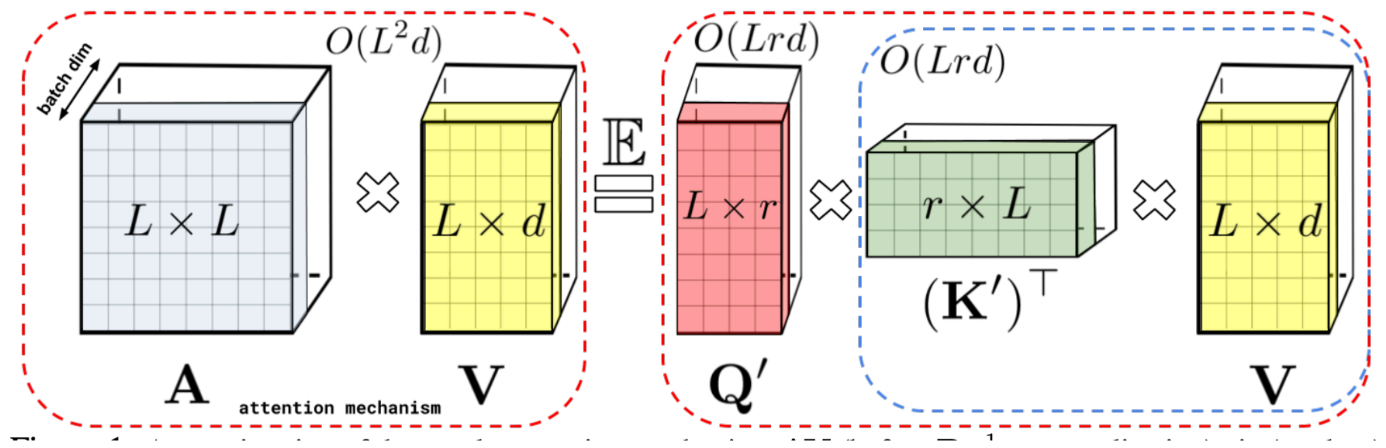
What about other architectures?

RNNs, LSTMs, Sub-quadratic attention…

Theorem [GM09]: A k-player r-round s-space protocol where players write blackboard messages in fixed order requires either $r \geq k$ or $s = \Omega(N/k^6)$ to solve hop_k.

➡**Multi-layer RNNs** require depth $L \geq k$ or memory $m = \Omega(N/k^6)$ to solve hop_k.

➡**Performers** require $L \geq k$ or $mm'H = \Omega(N/k^6)$ to solve hop_k.



- If $L, H = O(\log N)$, then only Performers with attention runtime $\tilde{\Omega}(N^2)$ can solve $\mathrm{hop}_{\log N}$.

Our results

Theorem 1: Transformers simulate MPC protocols.

↓

Log-depth Transformers can solve connected components & pointer chasing

* PC construction is empirically learnable by log-depth transformers!

Theorem 2: MPC protocols simulate transformers.

↓

Transformers require log-depth to solve CC & PC under MPC conjecture

* GNNs, RNNs, sub-quadratic attention models require poly-depth!

Thank you

References

[ACY23] Dana Angluin, David Chiang, Andy Yang. “Masked Hard-Attention Transformers and Boolean RASP Recognize Exactly the Star-Free Languages.” 2023.

[LAGKZ22] Bingbin Liu, Jordan T. Ash, Surbhi Goel, Akshay Krishnamurthy, Cyril Zhang. “Transformers Learn Shortcuts to Automata.” 2022.

[MSS22] William Merrill, Ashish Sabharwal, Noah Smith. “Saturated Transformers are Constant-Depth Threshold Circuits.” 2022.

[BCBJB23] Alberto Bietti, Viven Cabannes, Diane Bouchacourt, Hervé Jégou, Léon Bottou. “Birth of a Transformer: A Memory Viewpoint.” 2023.

[ASSWZ18] Alexandr Andoni, Zhao Song, Clifford Stein, Zhengyu Wang, Peilin Zhong. “Parallel Graph Connectivity and Log Diameter Rounds.” 2018.

[CLD+21] Krzysztof Choromanski, et al. “Rethinking Attention with Performers.” 2021.

[KSV10] Howard Karloff, Siddharth Suri, Sergei Vassilvitskii. A Model of Computation for MapReduce.” 2010.

[GKU19] Mohsen Ghaffari, Fabian Kuhn, and Jara Uitto. “Conditional hardness results for massively parallel computation from distributed lower bounds.” 2019.

[PMB19] Jorge Pérez, Javier Marinković, and Pablo Barceló. “On the Turing completeness of modern neural network architectures.” 2019.

[YBR+20] Chulhee Yun, Srinadh Bhojanapalli, Ankit Singh Rawat, Sashank Reddi, and Sanjiv Kumar. “Are transformers universal approximators of sequence-to-sequence functions?” ICLR 2020.

[WCM22] Colin Wei, Yining Chen, and Tengyu Ma. “Statistically meaningful approximation: a case study on approximating turing machines with transformers.” NeurIPS 2022.

[BAG20] Satwik Bhattamishra, Kabir Ahuja, and Navin Goyal. “On the ability and limitations of transformers to recognize formal languages.” EMNLP 2020.

[YPPN21] Shunyu Yao, Binghui Peng, Christos H. Papadimitriou, and Karthik Narasimhan. “Self-attention networks can process bounded hierarchical languages.” ACL 2021.

[LAG+22] Bingbin Liu, Jordan T. Ash, Surbhi Goel, Akshay Krishnamurthy, and Cyril Zhang. “Transformers learn shortcuts to automata.” 2022.

[HAF22] Yiding Hao, Dana Angluin, and Robert Frank. “Formal language recognition by hard attention transformers: Perspectives from circuit complexity.” 2022.

[EGKZ22] Benjamin L. Edelman, Surbhi Goel, Sham M. Kakade, and Cyril Zhang. “Inductive biases and variable creation in self-attention mechanisms.” ICML 2022.

[BPKP22] Satwik Bhattamishra, Arkil Patel, Varun Kanade, and Phil Blunsom. “Simplicity bias in transformers and their ability to learn sparse boolean functions.” 2022.

[ZFB23] Ruiqi Zhang, Spencer Frei, Peter Bartlett. “Trained Transformers Learn Linear Models In-Context.” 2023.

[Lou19] Andreas Loukas. “What graph neural networks cannot learn: depth vs width.” 2019.

[XHLG18] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. “How powerful are graph neural networks?” 2018.

[CBCB19] Zhengdao Chen, Soledad Villar, Lei Chen, and Joan Bruna. “On the equivalence between graph isomorphism testing and function approximation with GNNs.” NeurIPS 2019.

[MRF+19] Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. “Weisfeiler and leman go neural: Higher-order graph neural networks.” AAAI 2019.