A distributed computing lens on transformers

Clayton Sanford

April 9th, 2024

Columbia University

Google Research

Talk outline

1. [2 minutes] Introduction
2. [10 minutes] Overview of transformer architecture and theoretical results
3. [15 minutes] Equivalence between transformers and distributed computing
4. [10 minutes] Empirical study of powers of log-depth transformers
5. [10 minutes] Implications for sub-quadratic attention and state-space models
6. [5 minutes] Follow-up projects as an OMEGA SR

My research

Core goal: Understand neural net architectural trade-offs and inductive biases.

1. **Feedforward NN expressivity:**
   Effects of depth, weight-regularization, random features on representational powers.

2. **Learning low intrinsic-dimensional data:**
   Optimization results for two-layer NNs, analysis of inductive biases.

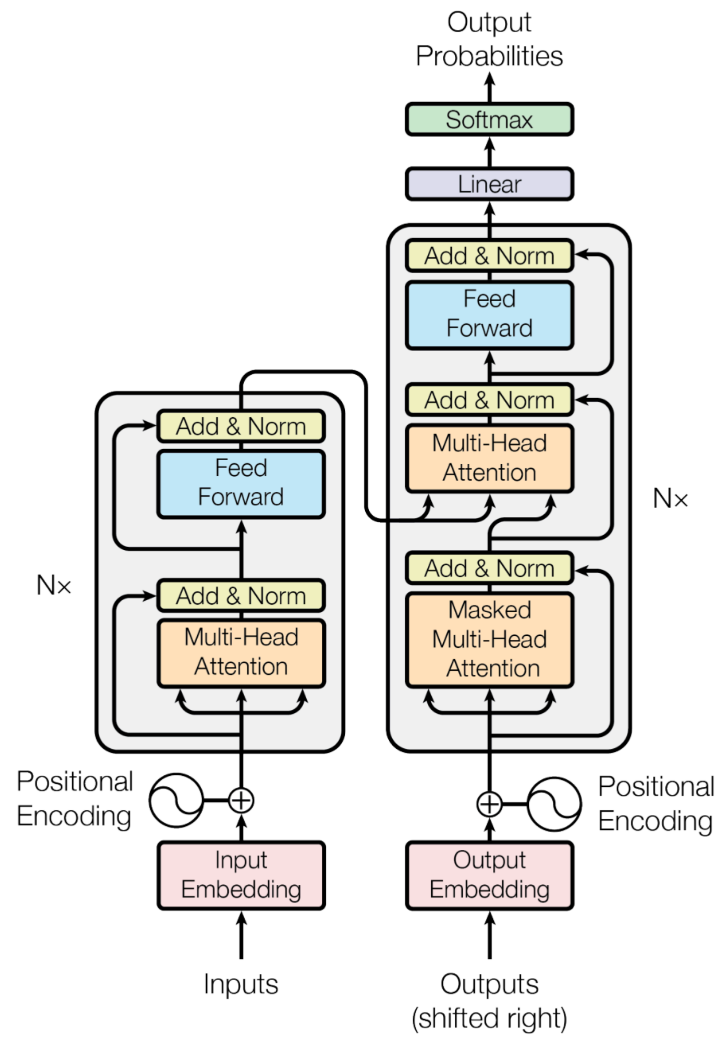
3. **Capabilities of sequential modeling architectures:**
   Effects of model size and architecture choice among transformers and state-space models.

4. **Interdisciplinary work:**
   Climate modeling + ML.

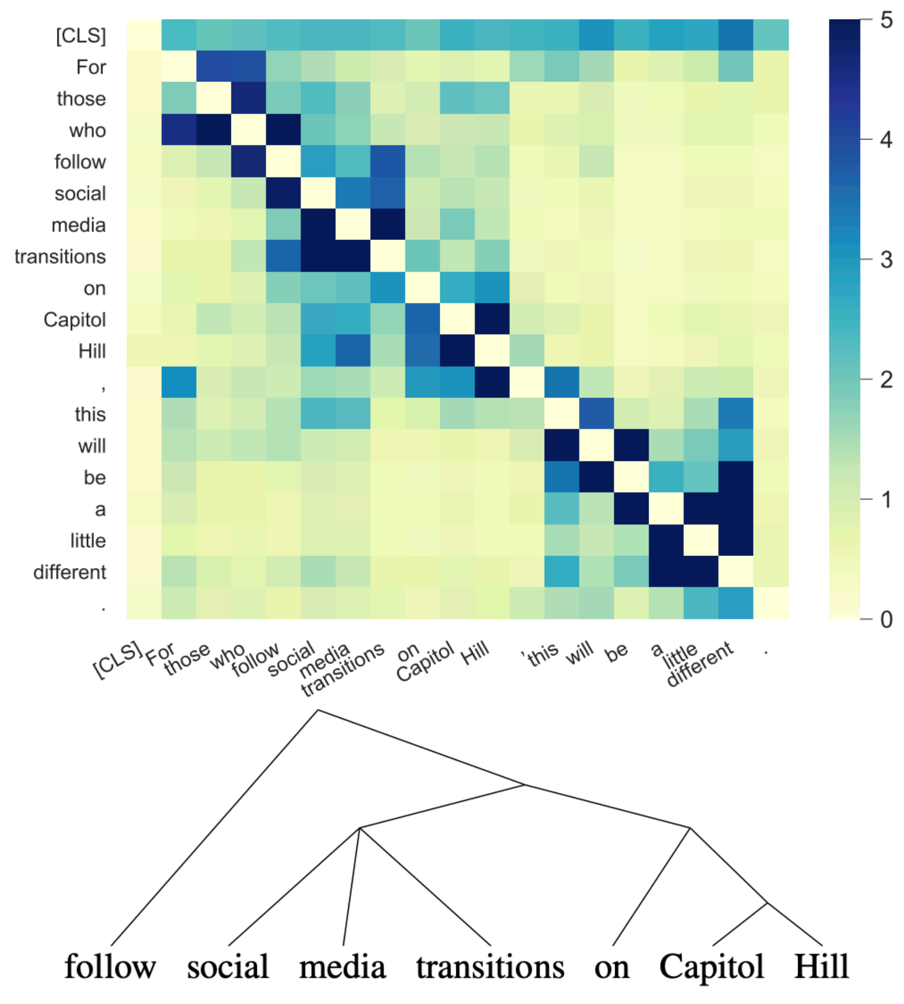
Transformers overview

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, 1M Gemini)
    - Quadratic computational bottleneck



Vaswani et al ‘17



Rogers et al ’20

Motivating questions

Practical questions

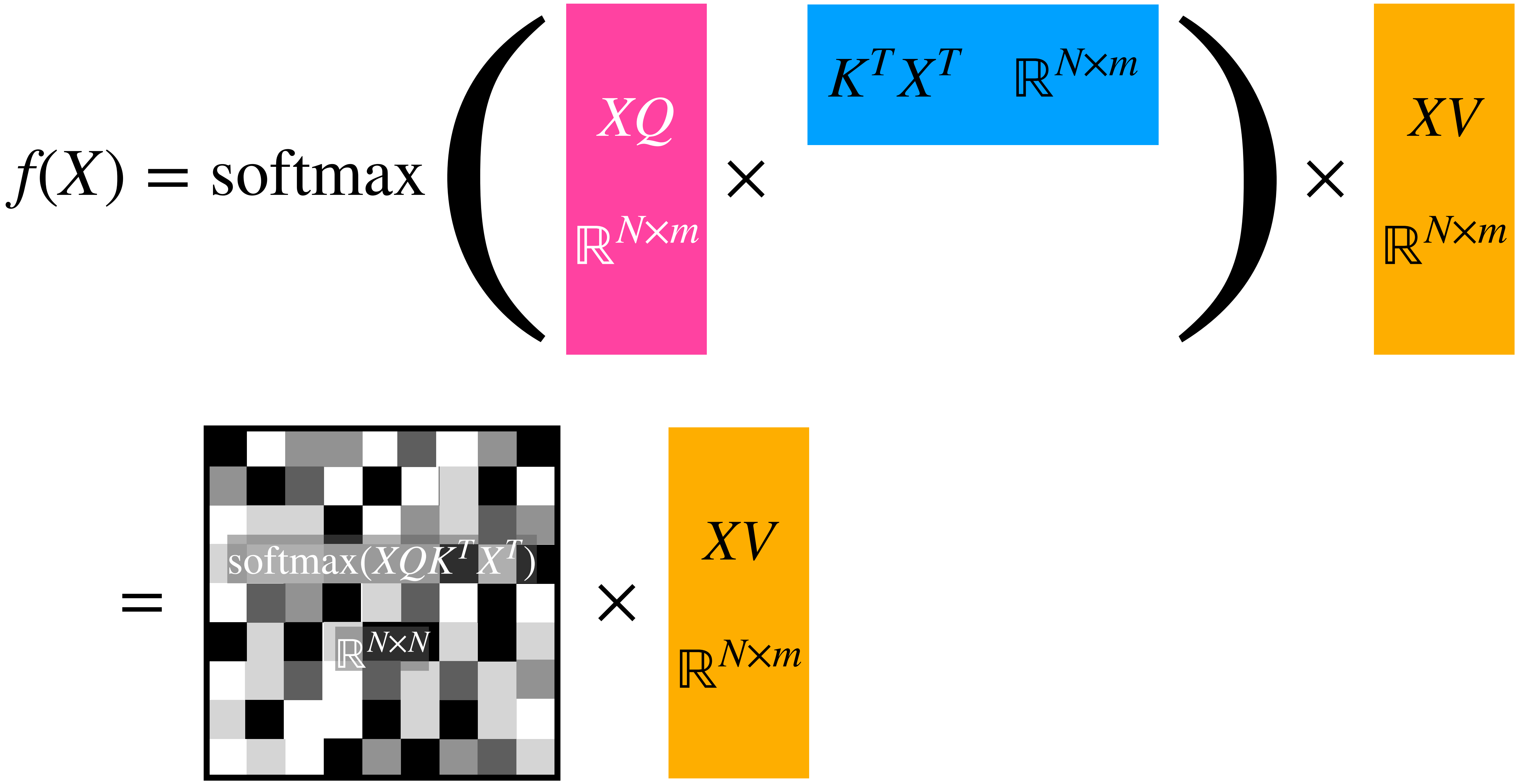
1. Are sub-quadratic attention models and state-space models (SSMs) as powerful as standard transformers?
2. Can transformers solve compositional tasks in a size-efficient manner?

Theoretical questions

1. Representational impacts of embedding dimensions and depth?
2. Separations with other models (RNNs, GNNs, sub-quadratic attention)?
3. Fundamental limitations of transformers?

Transformer architecture

- **Self-attention unit:**
  $f(X) = \mathrm{softmax}(XQK^TX^T)XV$
  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:**
  $T(X) = (\phi_L \circ g_L \circ \ldots \circ g_1 \circ \phi_0)(X)$



Our Contributions

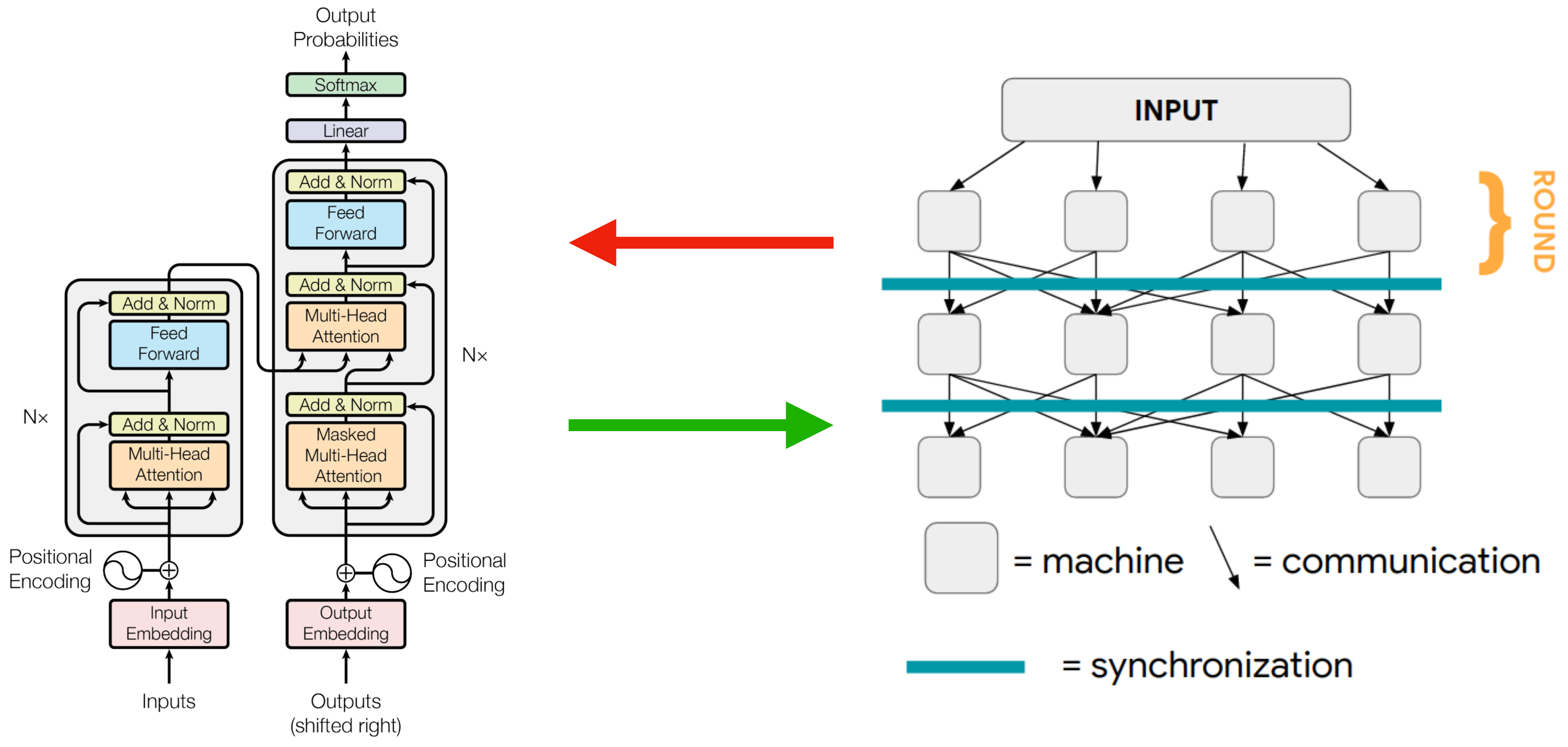
- 2-way relationship between transformers and **Massively Parallel Computation (MPC)** distributed computing model.

  ➡Transformers can implement parallelizable algorithms (theoretically and empirically).

  ➡Certain tasks require sufficient depth.

- Other models (SSMs, GNNs, some sub-quadratic models) correspond to serial “blackboard communication protocols.”

  ➡Separation between transformers and others on parallelizable tasks.

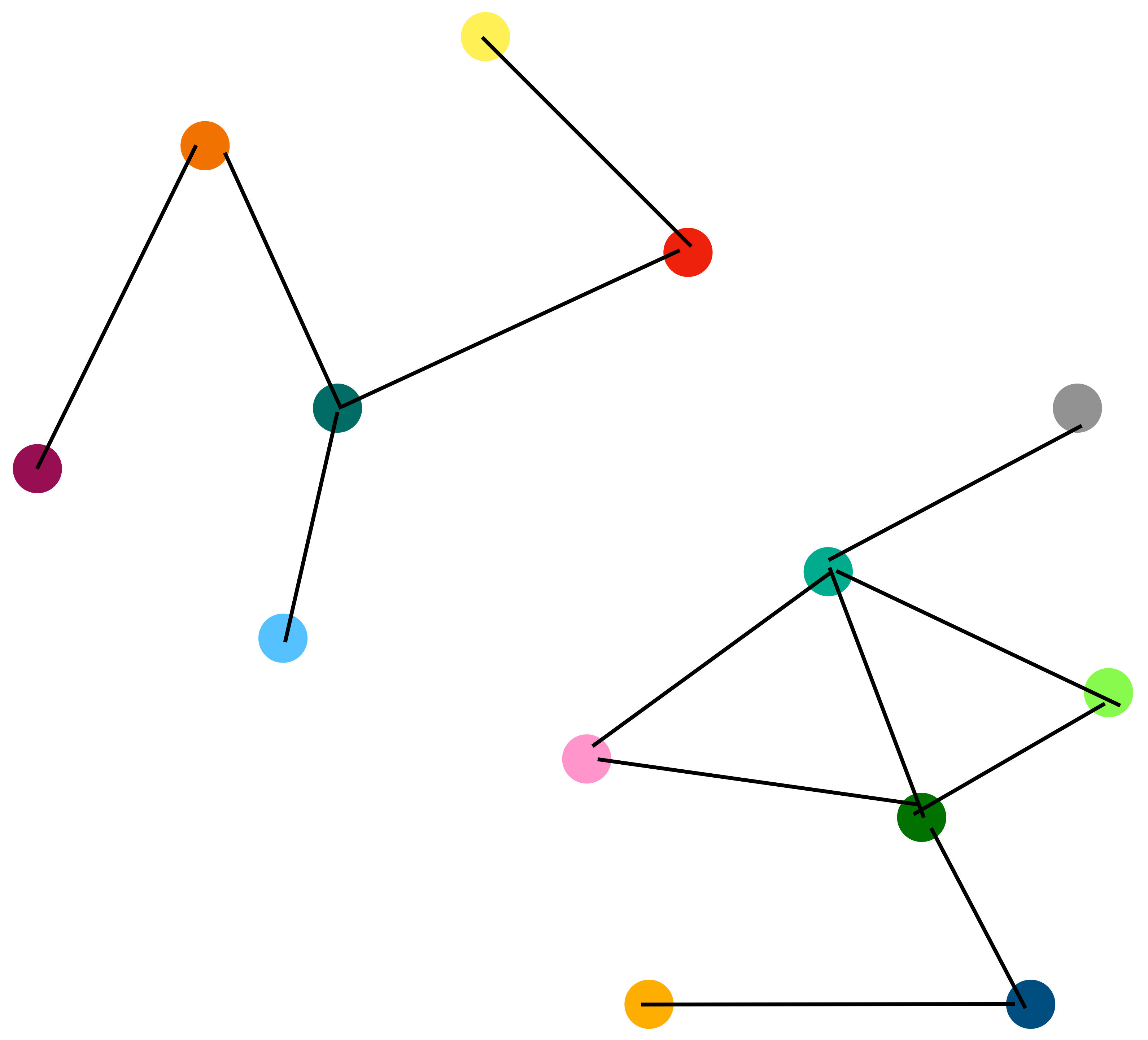


Transformers + Massively Parallel Computation

Takeaway: Computational equivalence between transformers and MPC

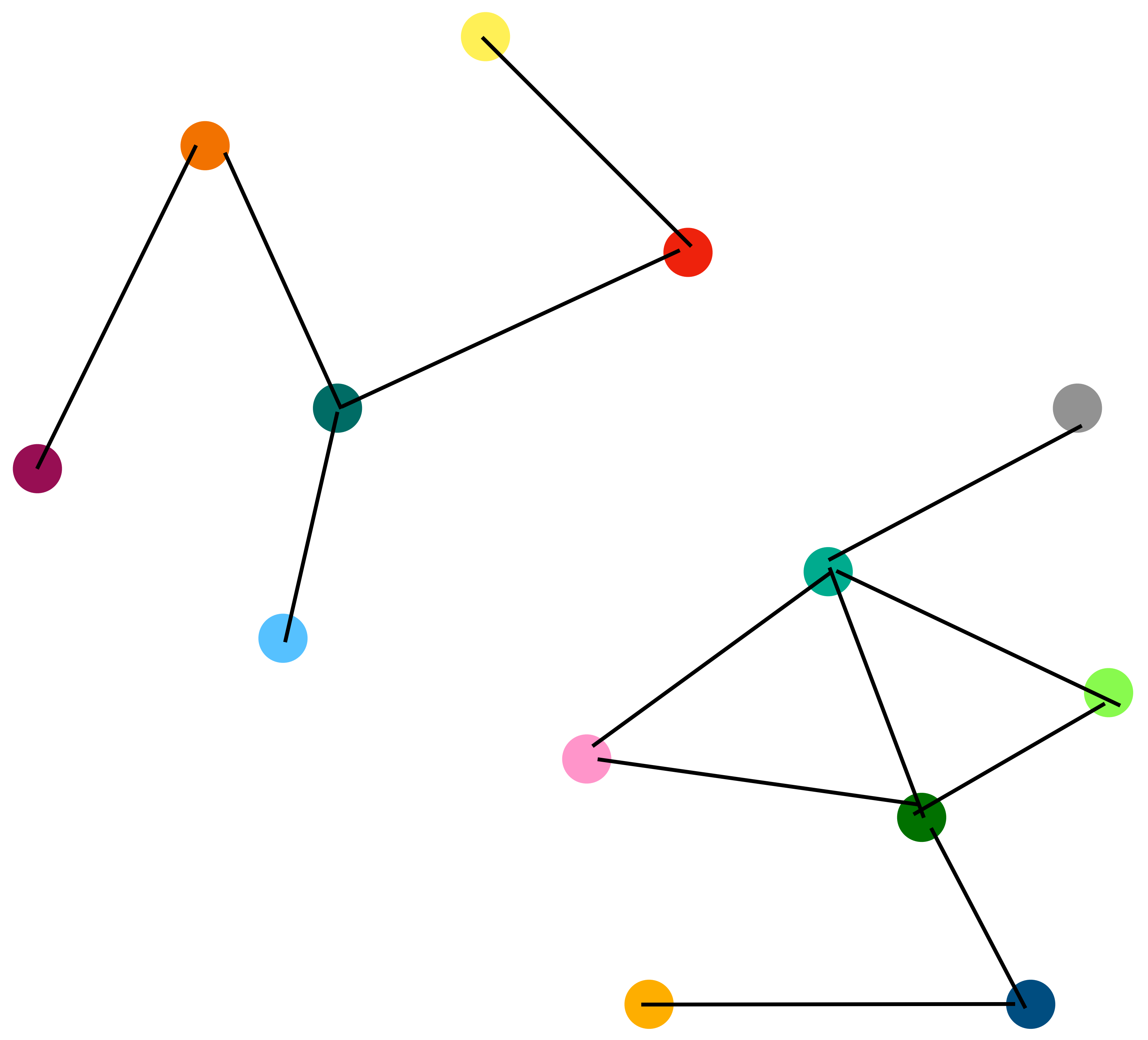
Parallelizable Tasks?

- Example: Graph connectivity
  - Given graph $G = (V, E)$, determine whether G is connected.
  - Solvable by DFS in $O(|V| + |E|)$ time.



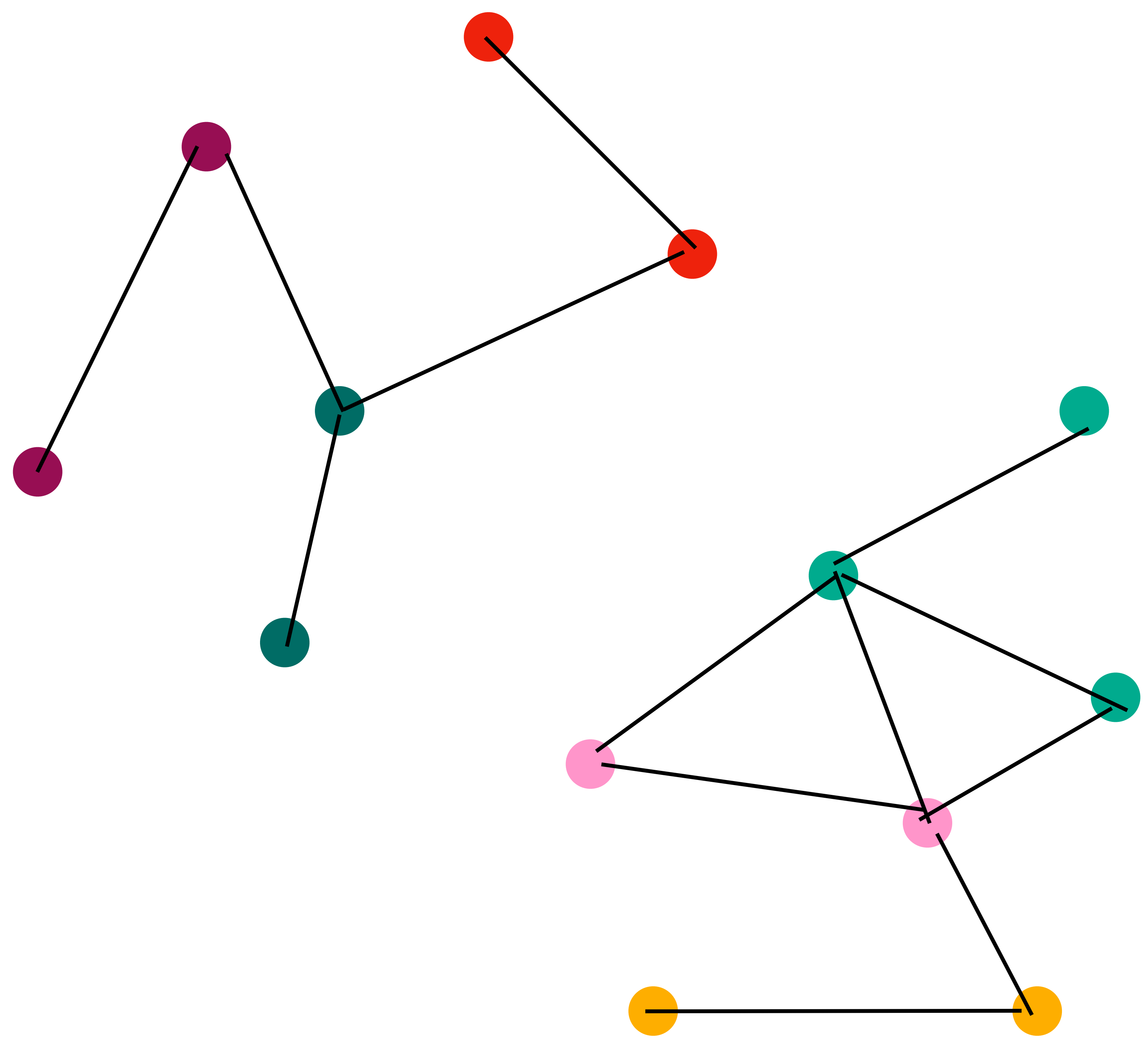
Parallelizable Tasks?

- Example: Graph connectivity
  - Given graph $G = (V, E)$, determine whether G is connected.
  - Solvable by DFS in $O(|V| + |E|)$ time.
  - Parallel algorithm in $O(\log |V|)$ rounds.



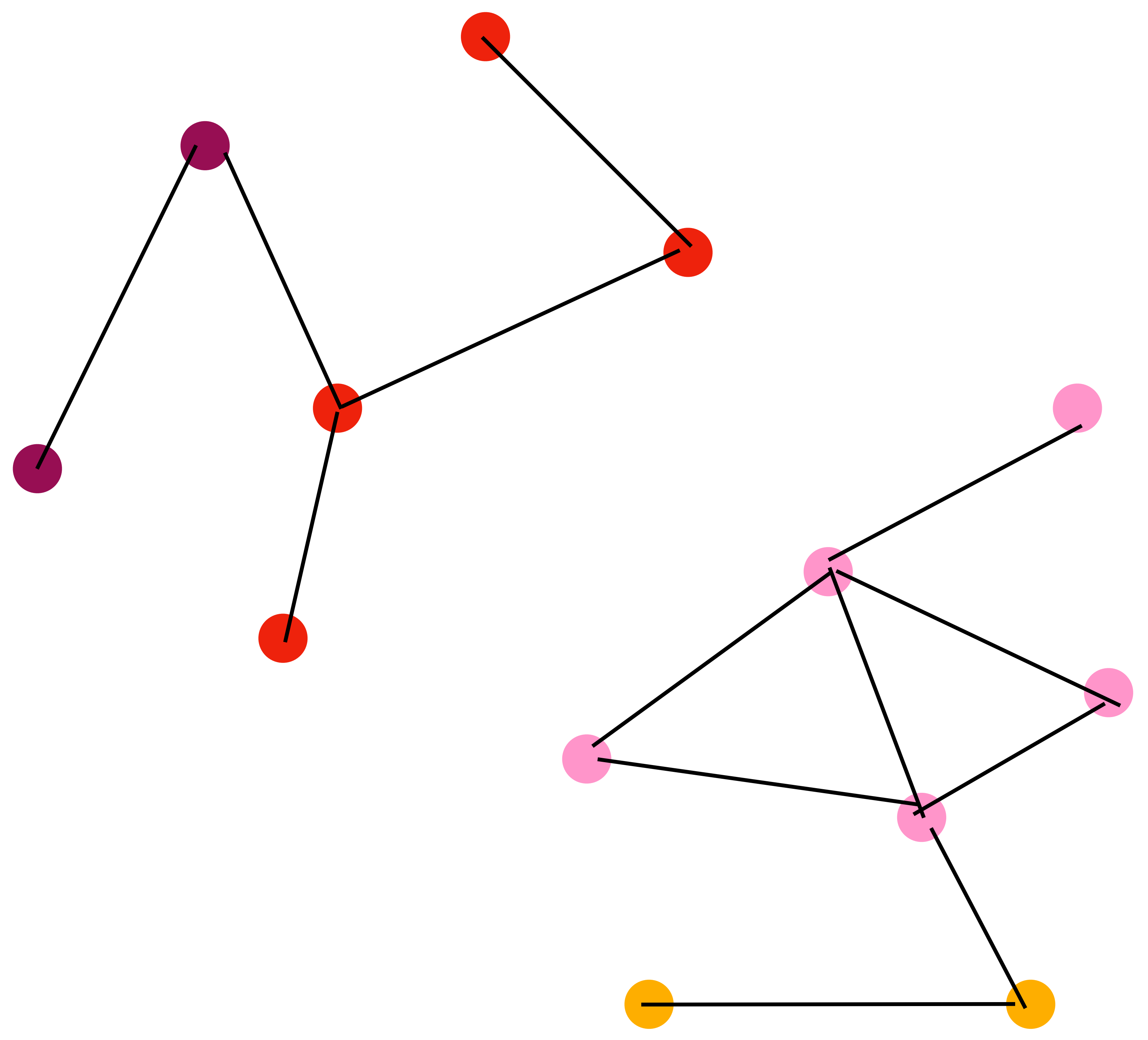
Parallelizable Tasks?

- Example: Graph connectivity
  - Given graph $G = (V, E)$, determine whether G is connected.
  - Solvable by DFS in $O(|V| + |E|)$ time.
  - Parallel algorithm in $O(\log |V|)$ rounds.



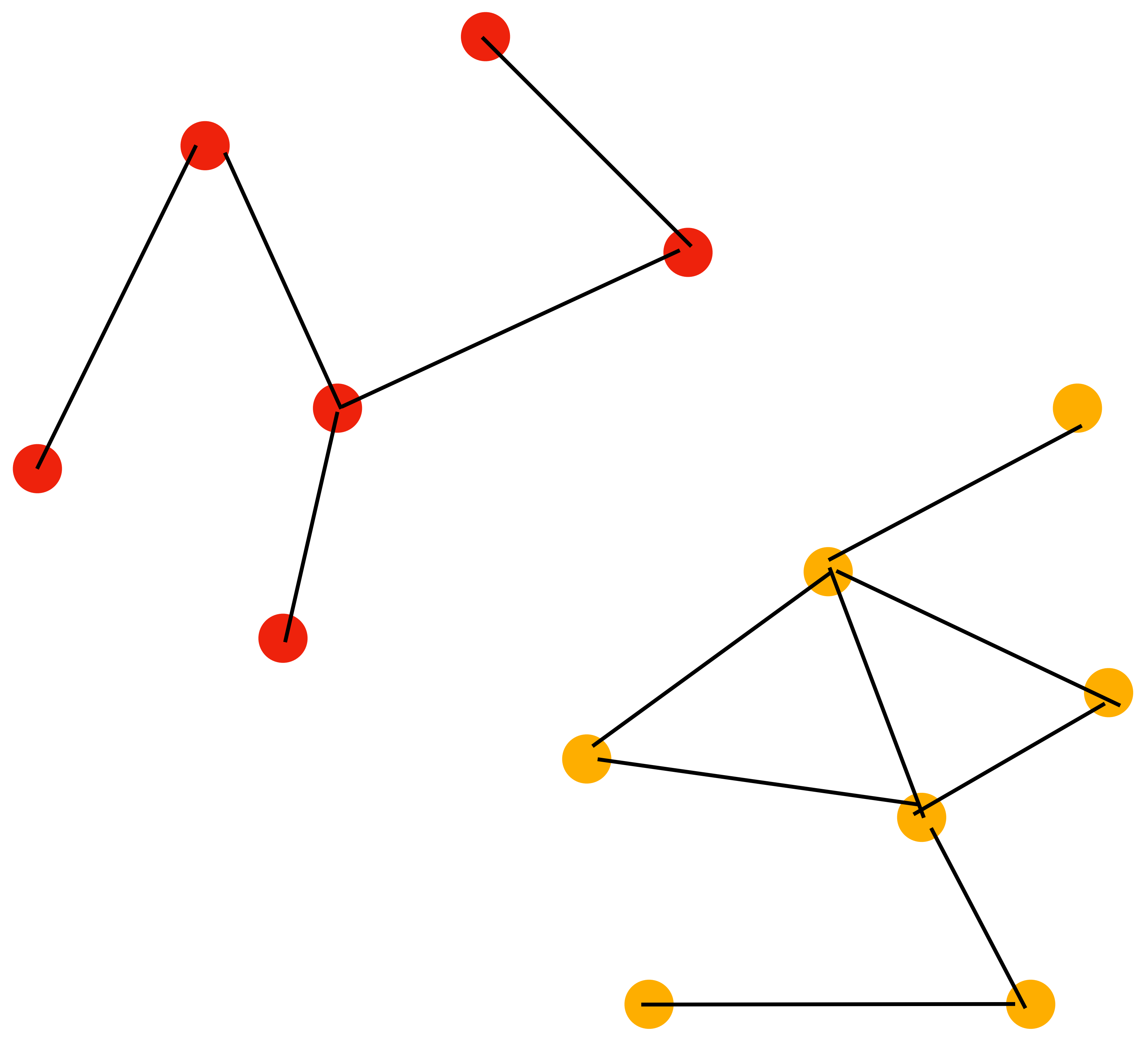
Parallelizable Tasks?

- Example: Graph connectivity
  - Given graph $G = (V, E)$, determine whether G is connected.
  - Solvable by DFS in $O(|V| + |E|)$ time.
  - Parallel algorithm in $O(\log |V|)$ rounds.



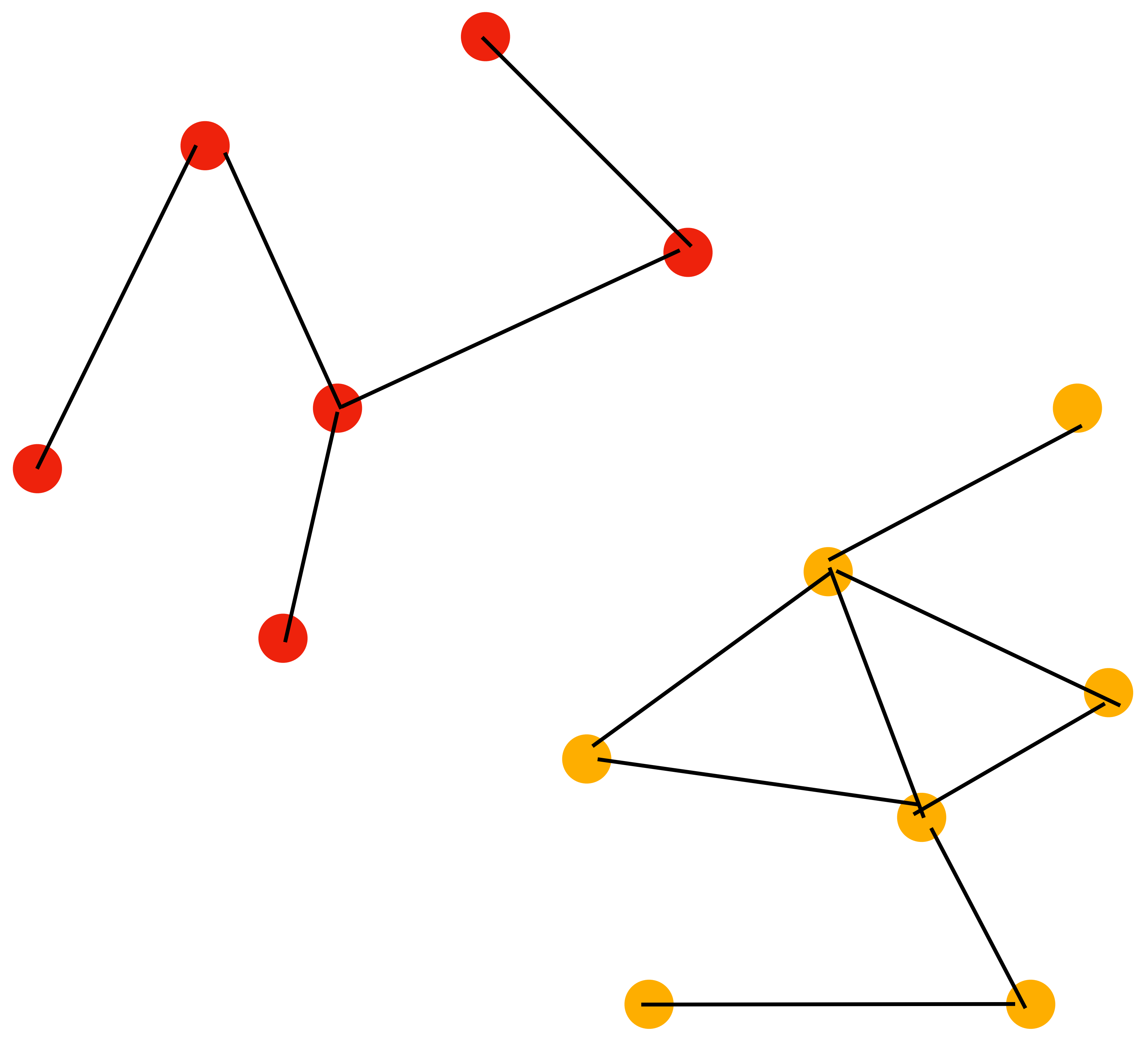
Parallelizable Tasks?

- Example: Graph connectivity
  - Given graph $G = (V, E)$, determine whether G is connected.
  - Solvable by DFS in $O(|V| + |E|)$ time.
  - Parallel algorithm in $O(\log |V|)$ rounds.



Parallelizable Tasks?

- Example: Graph connectivity
  - Given graph $G = (V, E)$, determine whether G is connected.
  - Solvable by DFS in $O(|V| + |E|)$ time.
  - Parallel algorithm in $O(\log |V|)$ rounds.
- Computational model for tasks solvable with algorithms of this form?



Massively Parallel Computation (MPC)

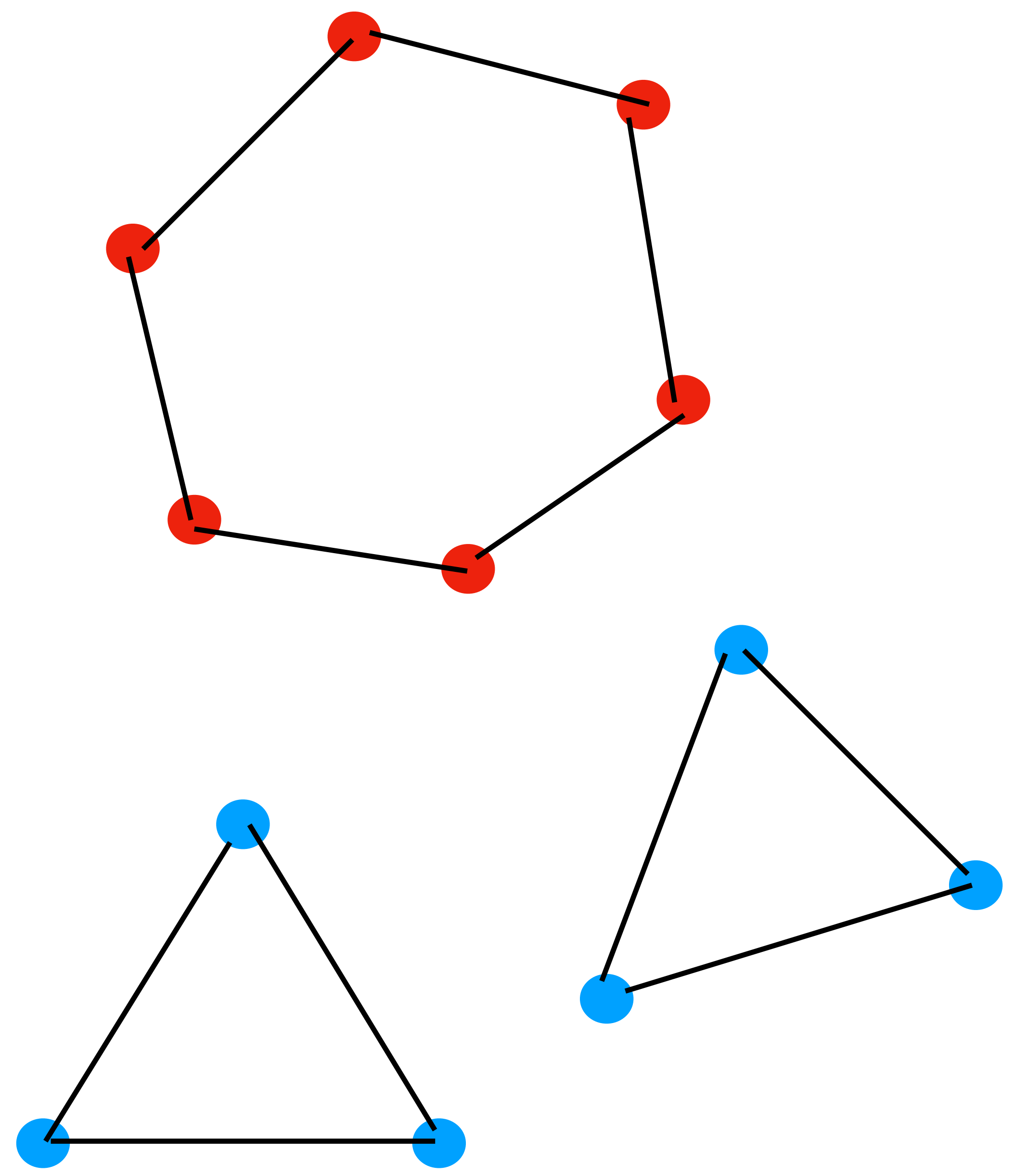
(not Multi-Party Computation)

- MPC = theoretical model of MapReduce
- A distributed protocol is MPC on size-N input of $O(\log N)$-bit words if:
  - q **machines**, each with **local memory** $s = N^{0.01}$.
  - **Global memory** $qs \leq N^{1.01}$.
  - Each of r rounds, machines perform parallel computation and send/receive messages simultaneously.
  - Unbounded computation, total size of messages sent and received per machine $\leq s$.

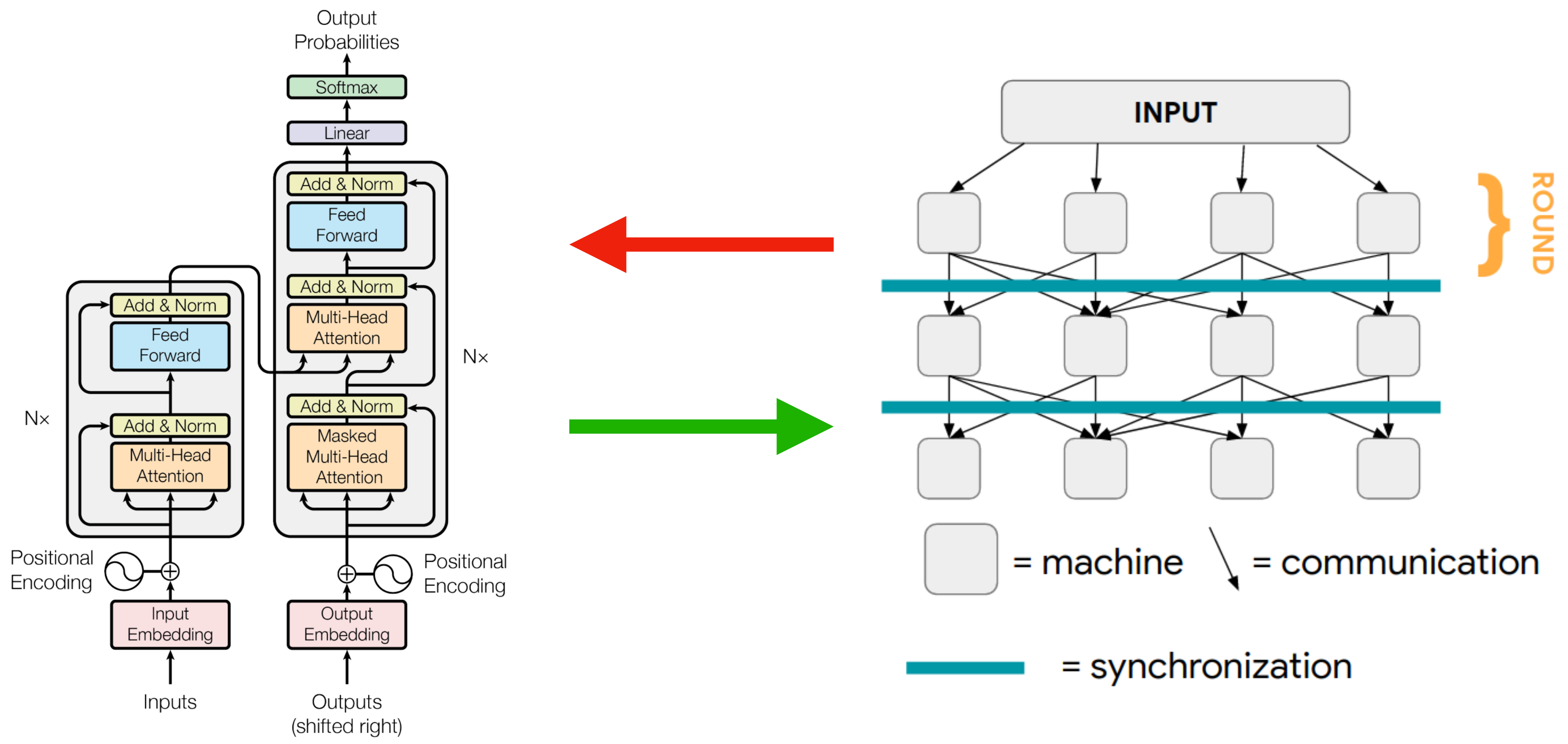
Massively Parallel Computation (MPC)

Examples [Andoni, Song, Stein, Wang, Zhong '18]

- Graph connectivity for $|E| = N$ is solvable in $R = O(\log N)$ rounds, $q = O(N)$ machines, $s = O(N^{0.01})$ local memory.
- Other graph problems: minimum spanning forest, diameter estimation.
- **One-cycle vs two-cycle conjecture**: Distinguishing cycle graphs of size N from two cycles of size $N/2$ with $q = \mathrm{poly}(N)$ machines and $s = o(N)$ local memory requires $R = \Omega(\log N)$ rounds.



Our results



Theorem 1: Transformers simulate MPC protocols.

Log-depth Transformers can solve connected components

Theorem 2: MPC protocols simulate transformers.

Transformers require log-depth to solve connected components under conjecture.

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

Simulating MPC with transformers

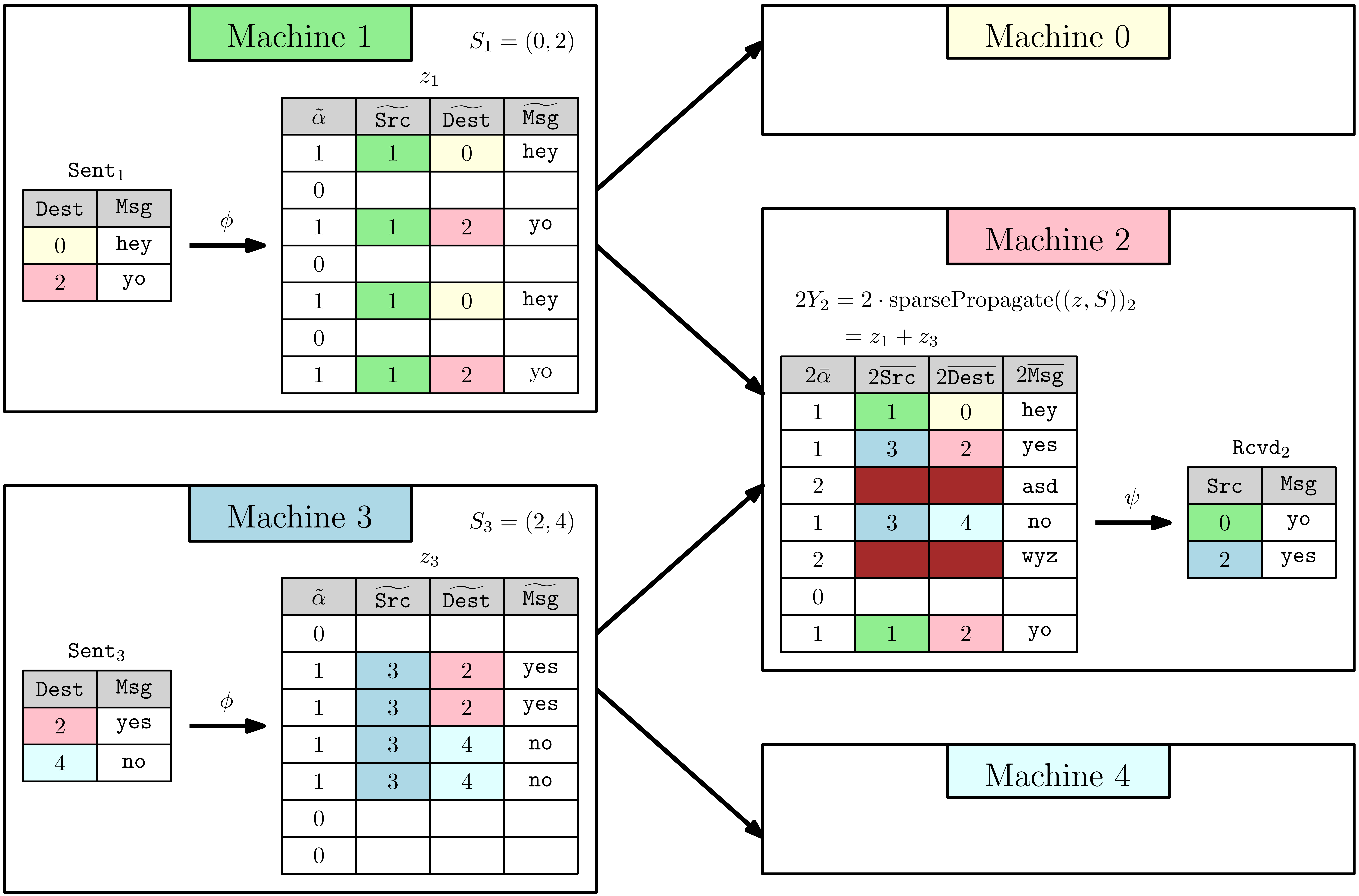
Theorem 1: Any R-round MPC protocol with $s = N^{\delta}$ local memory and $q \leq N$ machines can be simulated by a transformer with depth $L = O(R)$ and width $m = \tilde{O}(N^{4\delta})$.

- Proof idea:
  - Simulate local computation on in each MLP.
  - Route information between each machine using message passing-encoded self-attention layer.
    - Main technical challenge: Error correction via copies of messages in sparse locations on value vector XV.

Simulating MPC with transformers

Theorem 1: Any R-round MPC protocol with $s = N^{\delta}$ local memory and $q \leq N$ machines can be simulated by a transformer with depth $L = O(R)$ and width $m = \tilde{O}(N^{4\delta})$.

- Encode outgoing messages from each machine as repeated "packets" in value vector XV.
- Encode destination of messages in query vector XQ.
    - Sparse averaging results of [**S**, Hsu, Telgarsky '23] compute averages of all packets received by each input.
- Decode received average of value vectors, which is possible due to redundancy of packet structure.


Machine 1
$S_1 = (0, 2)$
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
$\texttt{Sent}_1$
Dest Msg
0 hey
2 yo
ϕ
Machine 3
$S_3 = (2, 4)$
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
$\texttt{Sent}_3$
Dest Msg
2 yes
4 no
ϕ
Machine 0
Machine 2
$2Y_2 = 2 \cdot \text{sparsePropagate}((z, S))_2$
$= z_1 + z_3$
$2\bar{\alpha}$ | $2\overline{\texttt{Src}}$ | $2\overline{\texttt{Dest}}$ | $2\overline{\texttt{Msg}}$
1 1 0 hey
1 3 2 yes
2 asd
1 3 4 no
2 wyz
0
1 1 2 yo
ψ
$\texttt{Rcvd}_2$
Src Msg
0 yo
2 yes
Machine 4

Simulating MPC with transformers

Theorem 1: Any R-round MPC protocol with $s = N^\delta$ local memory and $q \leq N$ machines can be simulated by a transformer with depth $L = O(R)$ and width $m = \tilde{O}(N^{\delta+0.0001})$.

- Improvement while at Google Research!

Transformers solve parallelizable algorithms

Theorem 1: Any R-round MPC protocol with $s = N^{\delta}$ local memory and $q \leq N$ machines can be simulated by a transformer with depth $L = O(R)$ and width $m = \tilde{O}(N^{\delta+0.0001})$.

Problem	MPC	Transformer
Graph connectivity	$s = \mathrm{O}(N^{0.01})$, $R = \mathrm{O}(\log N)$.	$L = \mathrm{O}(\log N)$, $m = \mathrm{O}(N^{0.02})$.
Min spanning forest	$s = \mathrm{O}(N^{0.01})$, $R = \mathrm{O}(\log N)$.	$L = \mathrm{O}(\log N)$, $m = \mathrm{O}(N^{0.02})$.
L **or** NL **Problems**	$s = \mathrm{O}(N^{0.51})$, $R = \mathrm{O}(\log N)$.	$L = \mathrm{O}(\log N)$, $m = \mathrm{O}(N^{0.52})$.

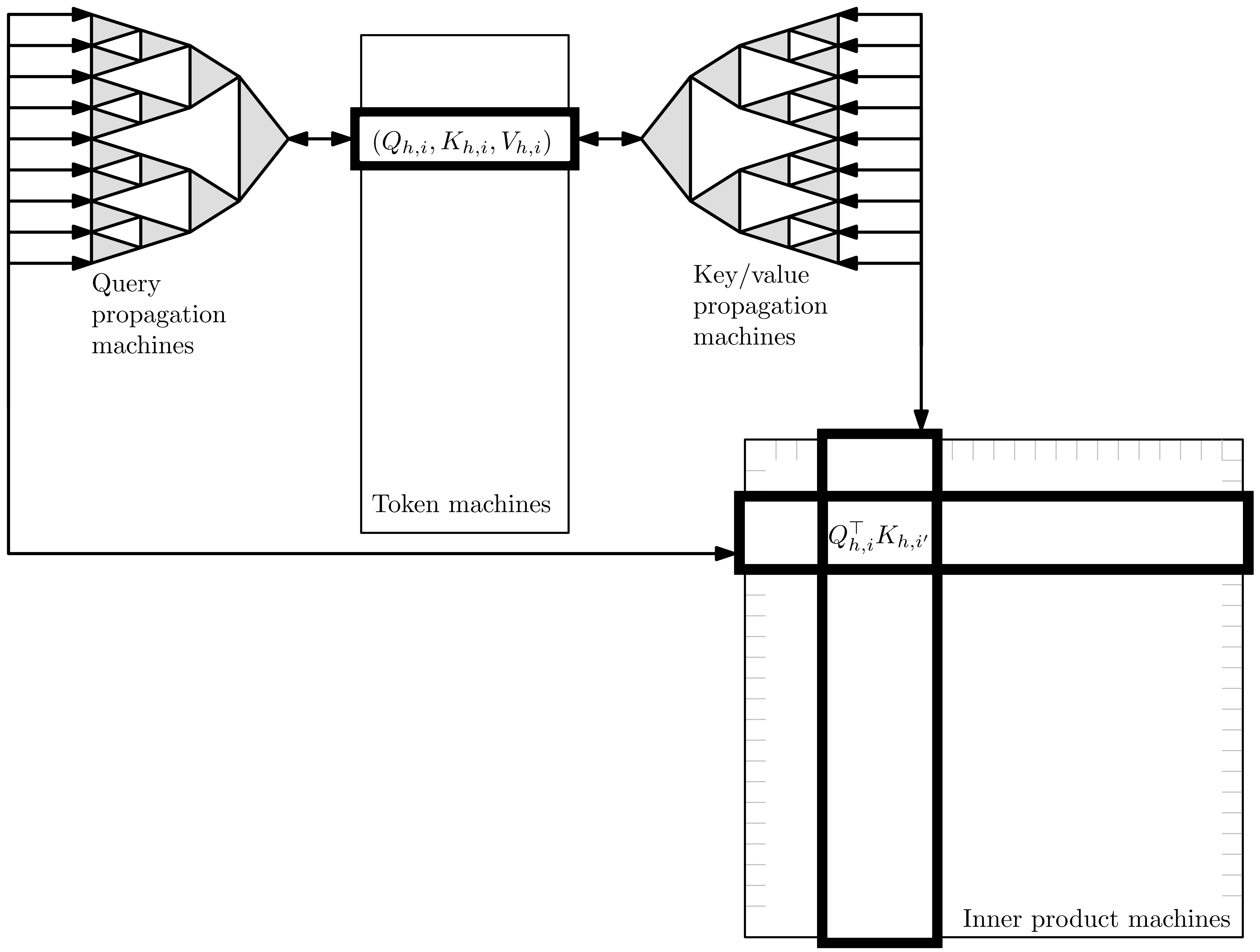
Simulating transformers with MPC

Theorem 2: Any transformer with depth L and width $m = N^\delta$ can be simulated by an $O(L/\delta)$-round MPC protocol with $q = O(N^2)$ machines and s = $N^{2\delta}$ local memory.

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

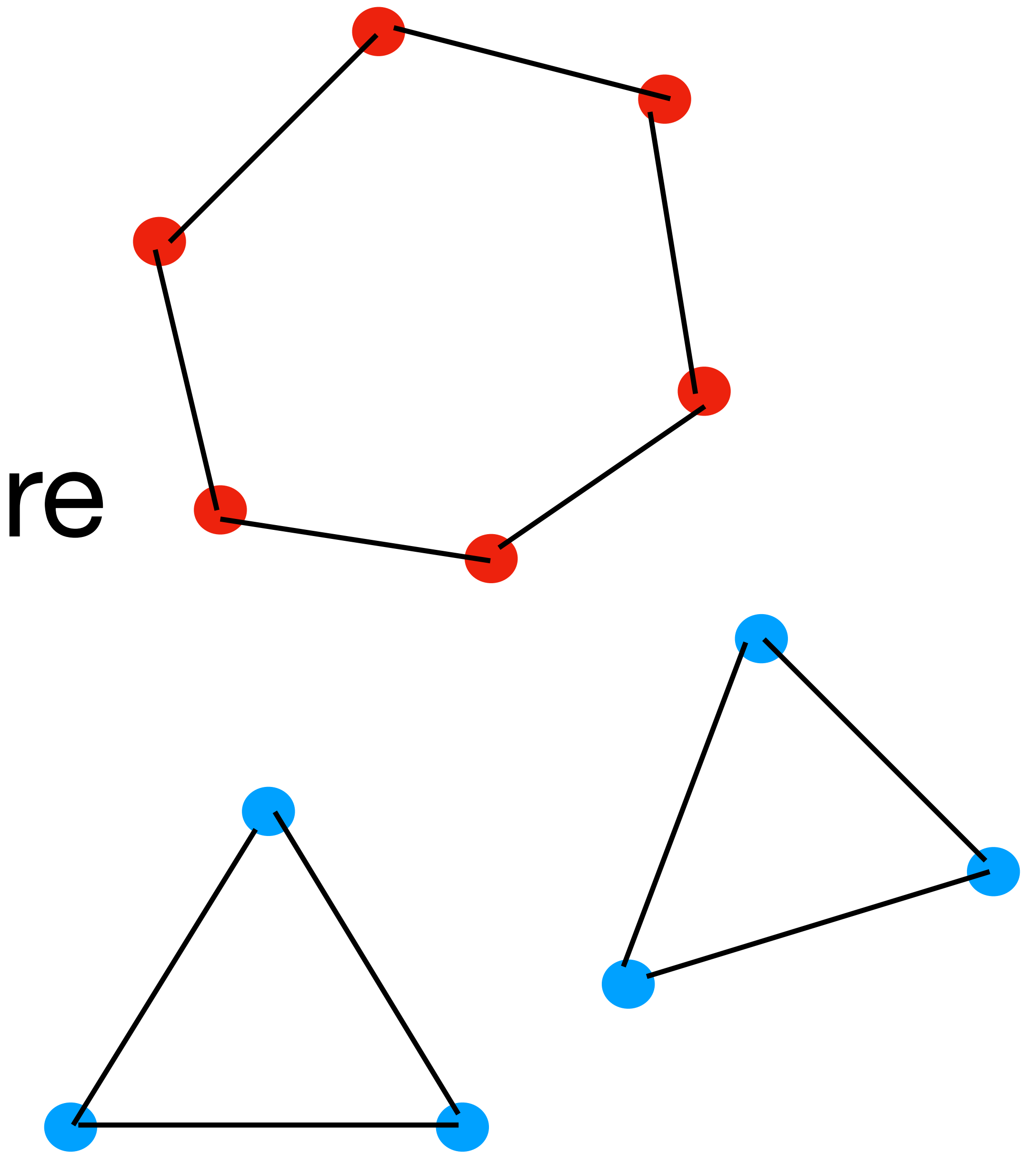
Simulating transformers with MPC

Theorem 2: Any transformer with depth L and width $m = N^{\delta}$ can be simulated by an $O(L/\delta)$-round MPC protocol with $q = O(N^2)$ machines and s = $N^{2\delta}$ local memory.



Optimality of parallel implementation

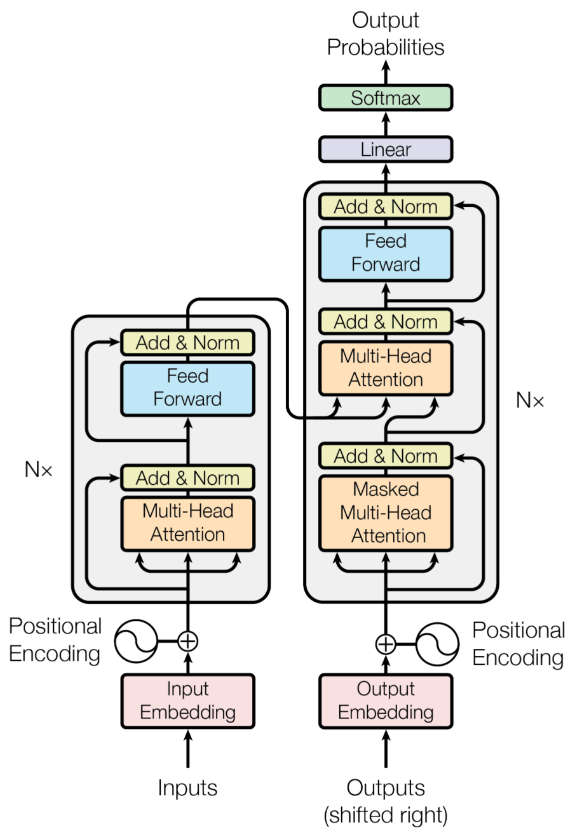
Theorem 2: Any transformer with depth L and width $m = N^\delta$ can be simulated by an $O(L/\delta)$-round MPC protocol with $q = O(N^2)$ machines and s = $N^{2\delta}$ local memory.

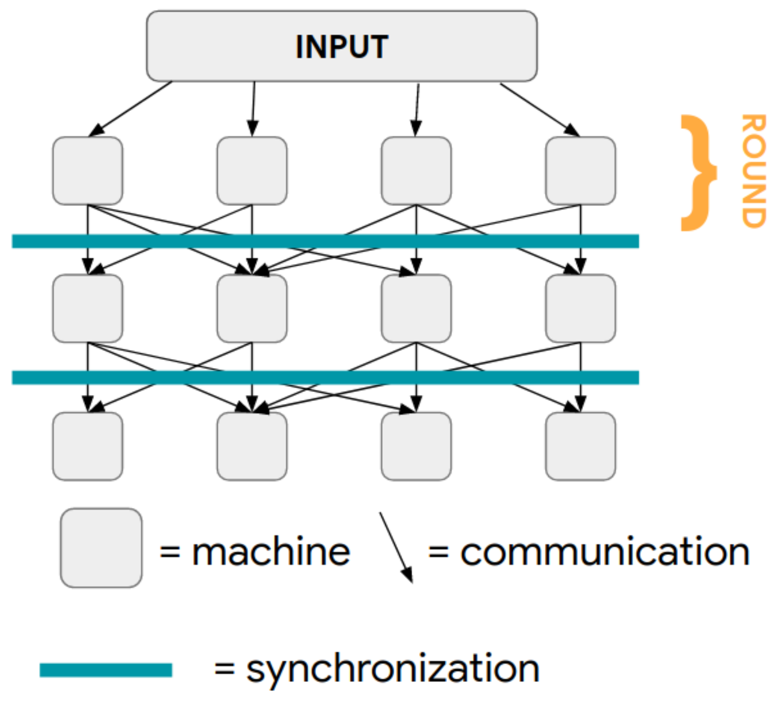


- Assuming 1-cycle vs 2-cycle conjecture:

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

Our results





Theorem 1: Transformers simulate MPC protocols.

Log-depth Transformers can solve connected components

Theorem 2: MPC protocols simulate transformers.

Transformers require log-depth to solve connected components under 1-cycle vs 2-cycle conjecture.

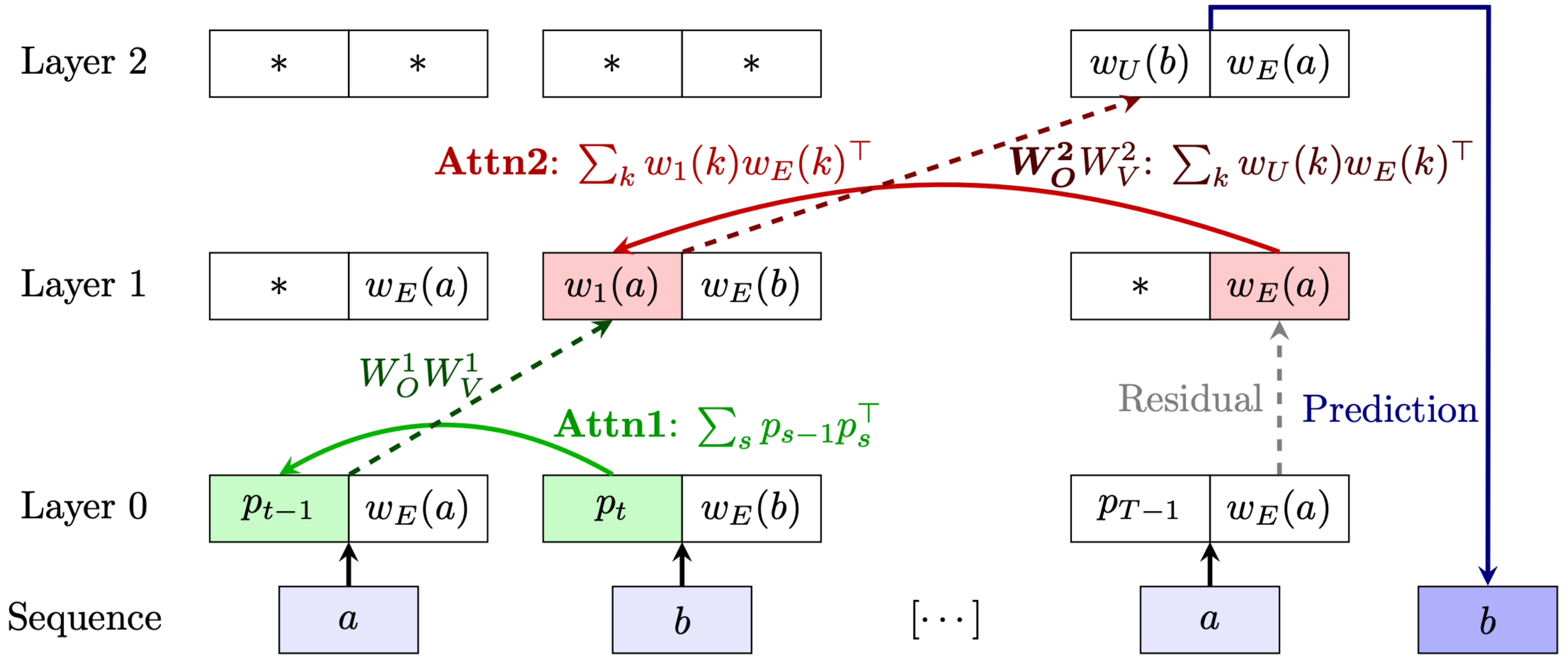
* GNNs, RNNs, sub-quadratic attention models require poly-depth!

Empirics + Mechanistic Interpretability

Takeaway: Parallelizable theoretical constructions are learnable and interpretable

Induction heads and powers of depth

- **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 [Bietti et al, '23]:
  - Layer 1: identify previous token
  - Layer 2: find most recent occurrence of token
- Impossible with 1 layer
  - Simple communication complexity proof



[BCBJB23]

Multi-hop induction heads and powers of depth

Task: k-hop induction heads:

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

baebcabebdea.

Transformer theory results

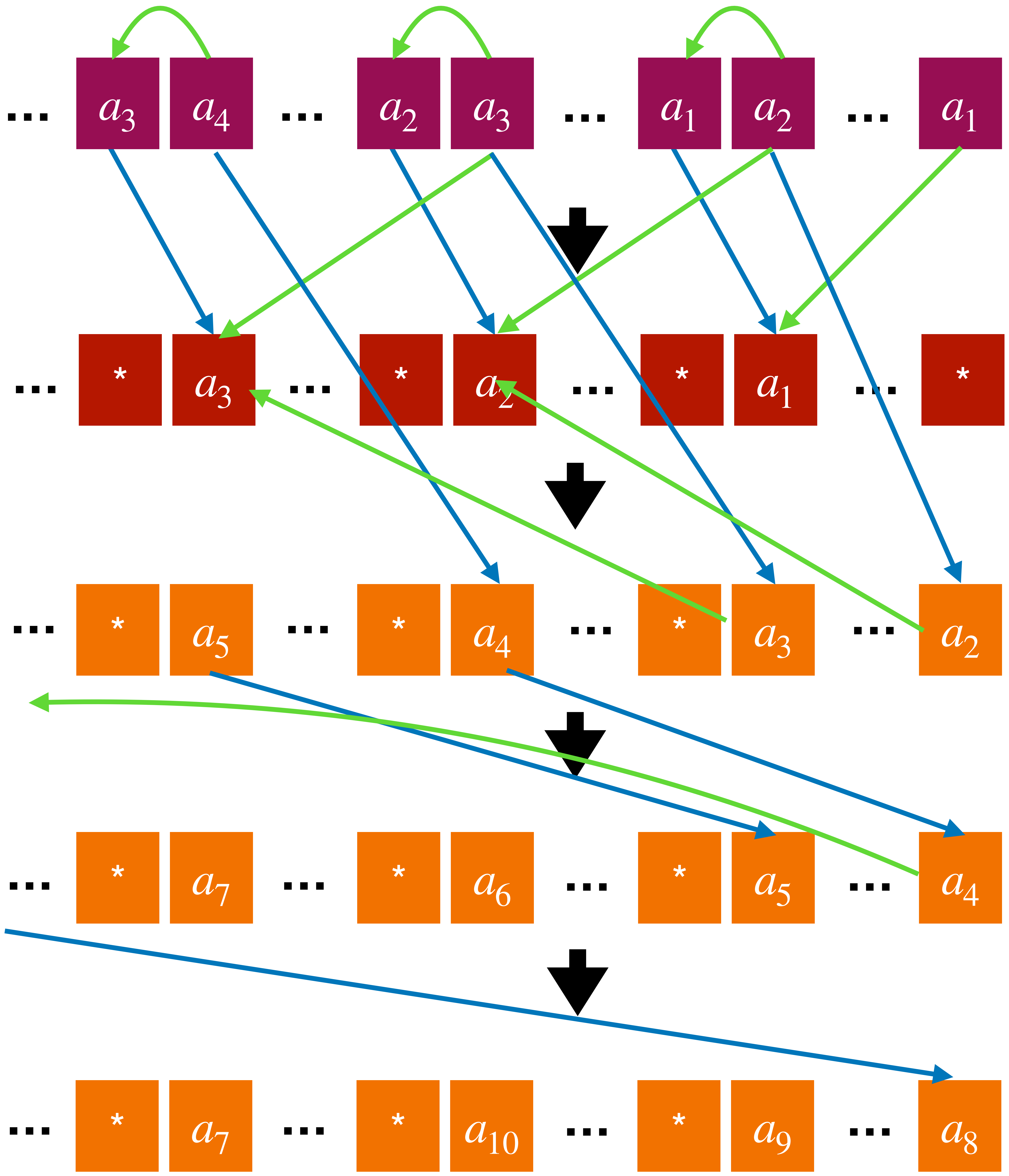
1. $O(\log k)$-depth constructions.
2. $\Omega(\log k)$ depth lower bounds (conditional).
3. Separation from other architectures (GNN, multi-layer RNNs, some sub-quadratic-attention transformers).

$\log(k)$-depth construction

Task: k-hop induction heads:

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

Theorem: There exists a transformer with depth $L = 2 + \log k$ and width $m = O(1)$ computing hop_k.



Learnability with gradient descent

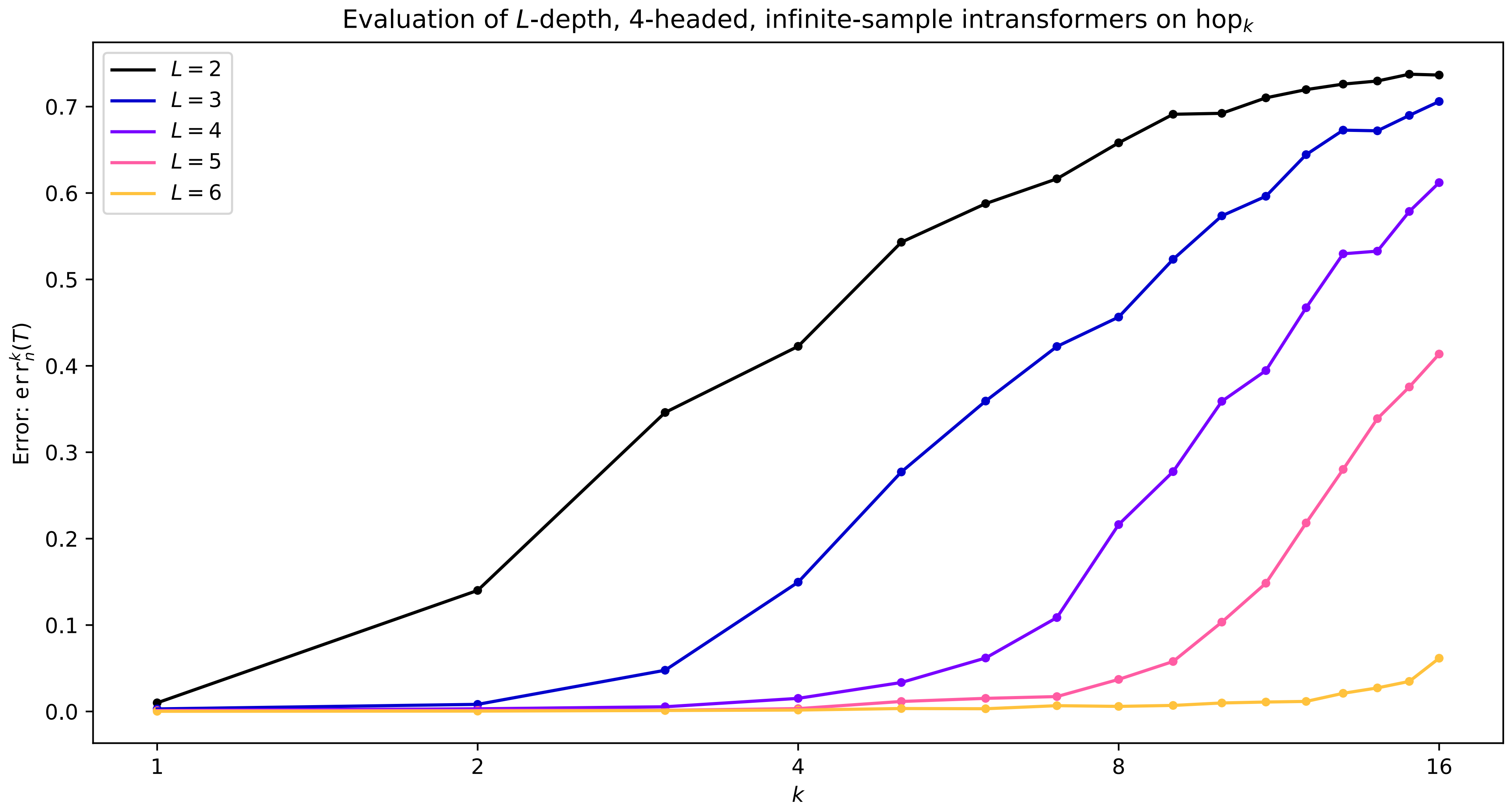
Task: k-hop induction heads:

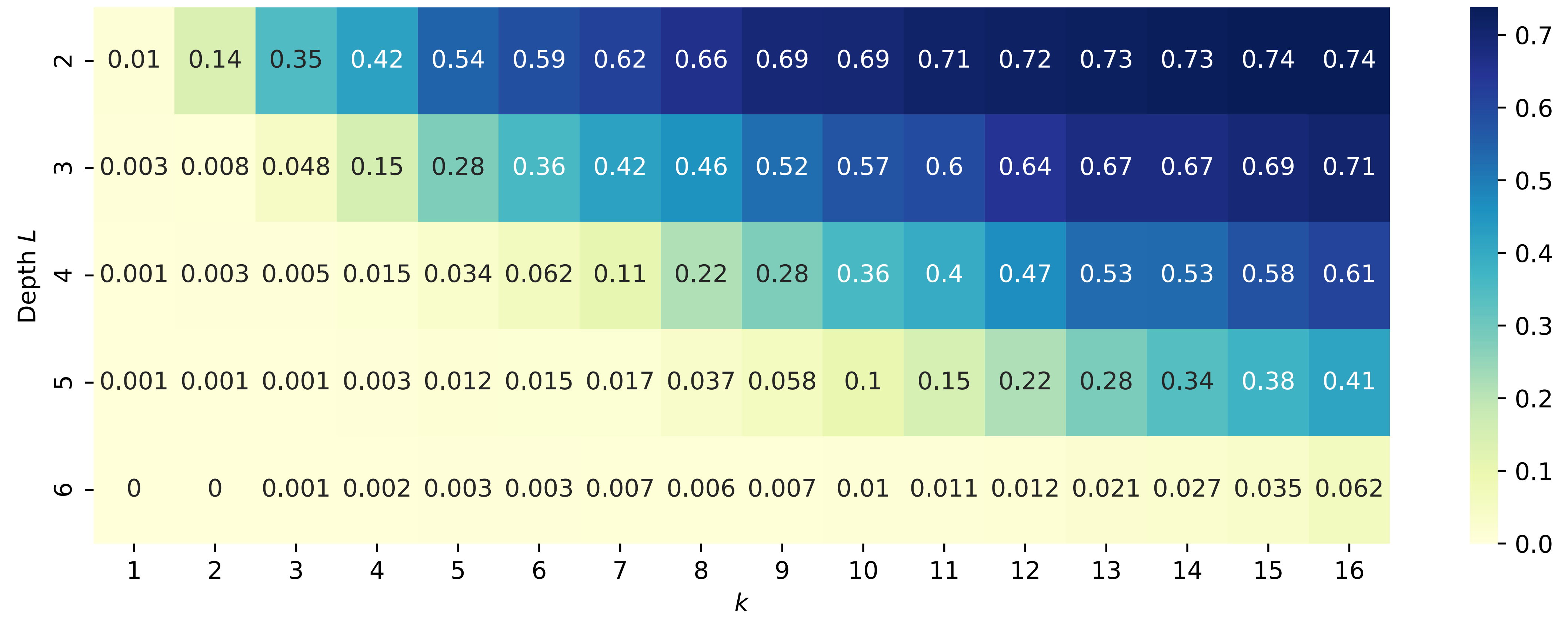
$$\text{hop}_k(\ldots a_k a_{k+1} \ldots a_{k-1} a_k \ldots a_1 a_2 \ldots a_1) = a_{k+1}$$

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

Task: k-hop induction heads:

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

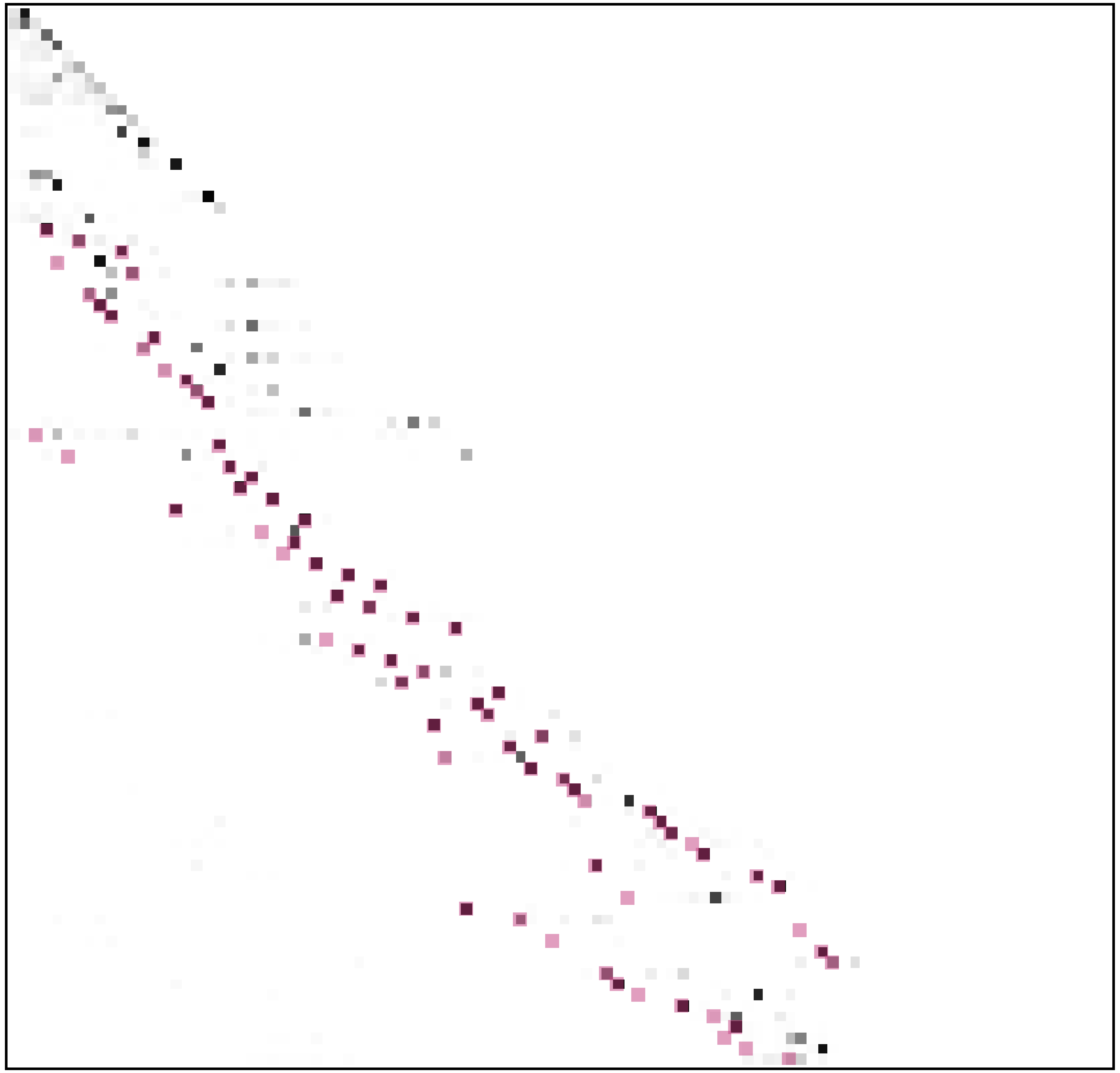
Empirical setting:

- 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:

Attention matrix: hop_{16}, $L = 6$, layer 6, head 1

Highlighting: hop_8 indices



Learnability with gradient descent

Task: k-hop induction heads:

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

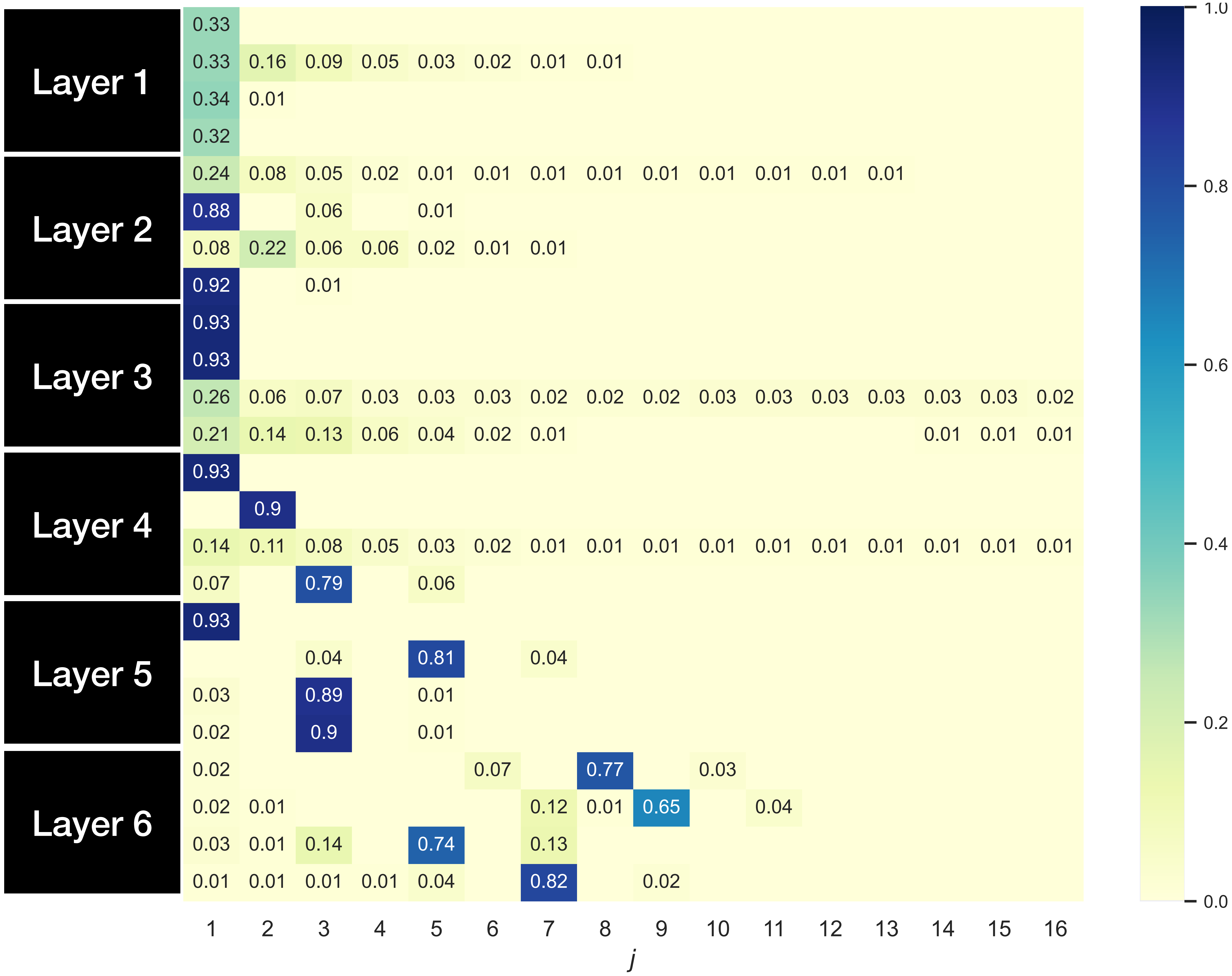
Empirical setting:

- 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

Attention matrix: hop_{16}, $L = 6$

Inner products with hop_j indices



Limitations of Alternative Architectures

Takeaway: Other architectures relate to other distributed computing models, which limits their capacity

Graph Neural Networks (GNNs) + CONGEST

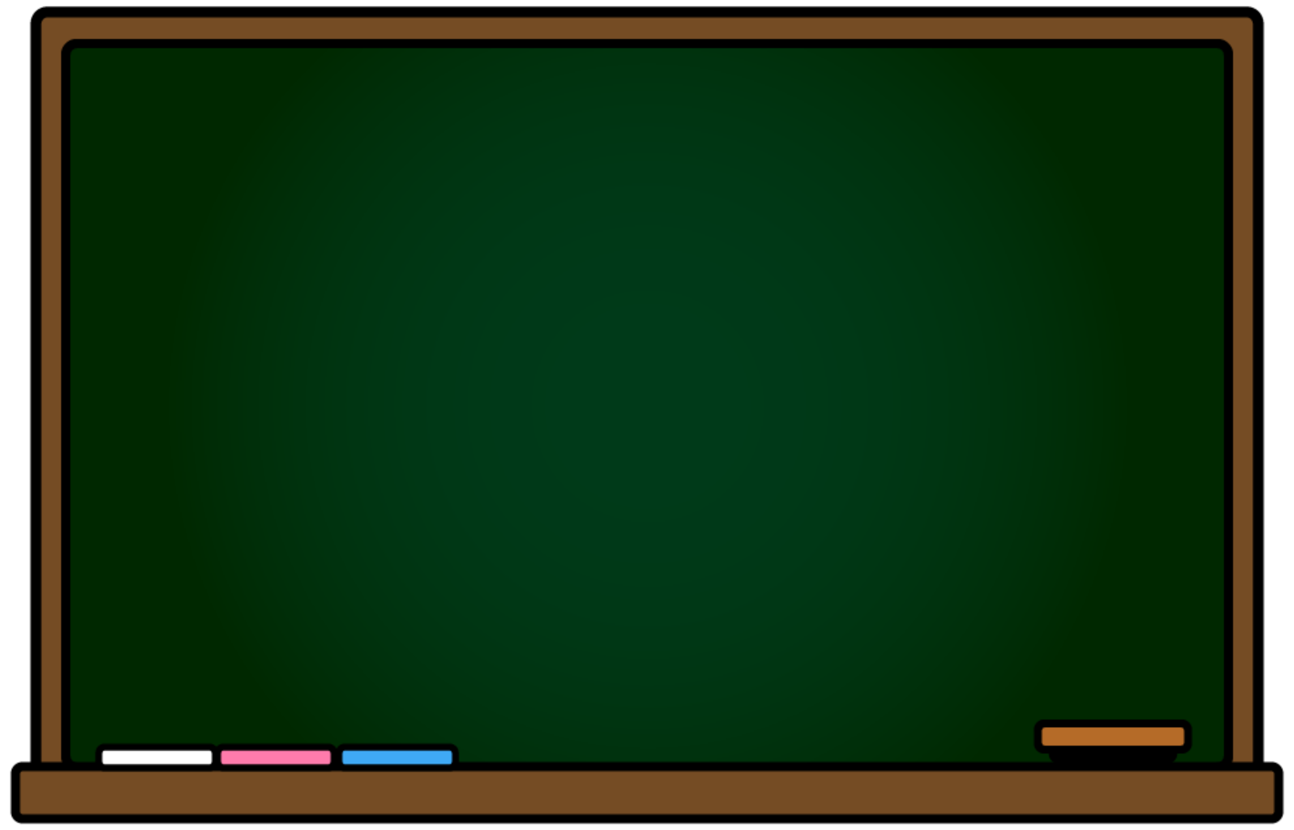
- [Loukas ’19] relates GNNs to CONGEST distributed computing model.
  - CONGEST: Fixed communication graph, each node sends simultaneous $O(\log N)$-bit messages to neighbors.
  - Bounded message size GNNs can be simulated by CONGEST.

  ➡GNNs require $L\sqrt{m} = \tilde{\Omega}(\sqrt{N})$ for subgraph connectivity.

  - vs depth $L = \log(N)$ and width $m = N^{0.01}$ for transformers.

Blackboard communication protocols

- "Bounded-communication parallel model"
  - k players $P_1, \ldots, P_k$ each receive N/k input tokens.
  - In each round, each player writes s bits of information to the blackboard in the order $P_1, \ldots, P_k$.
  - After r rounds, P_1 returns an answer.



- **Theorem [GM09]:** A blackboard protocol that solves hop_k requires $r \geq k$ or $s = \Omega(N/k^6)$.
  - Standard pointer chasing lower bound.
  - Think: $k = N^{0.01}$.

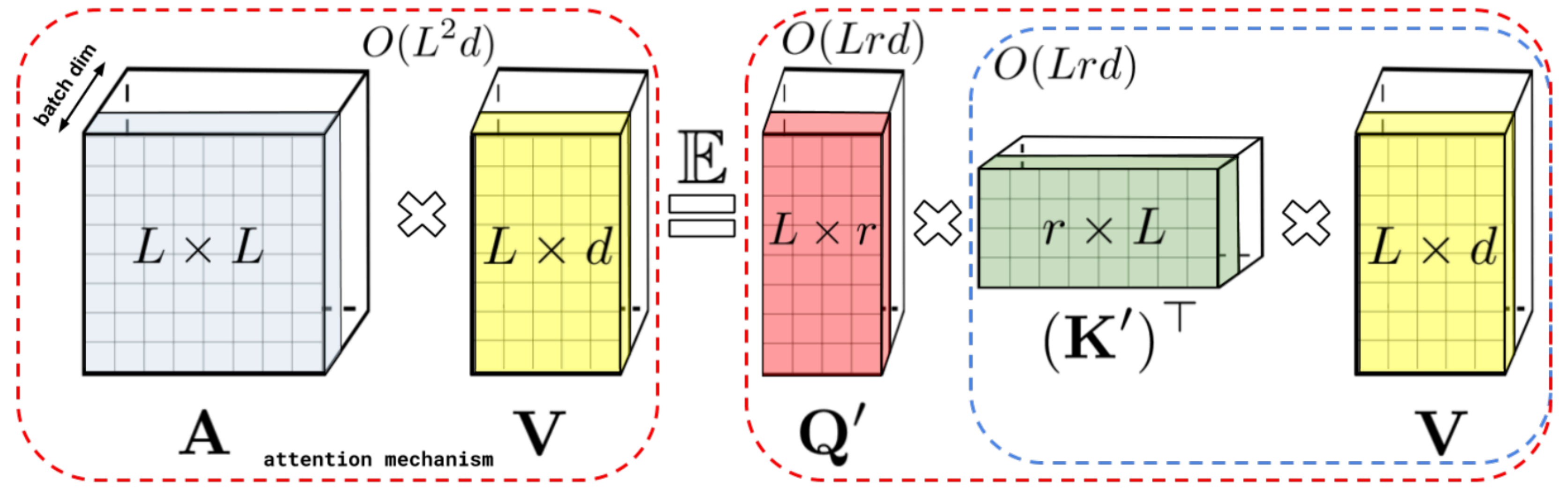
State-space models and linear-time attention

- L-layer m-width multi-pass RNNs can be simulated by an L-round m-size protocol.

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

- L-layer, m-embed. dim., H-head, r-feature dim. Performers can be simulated by an L-round $mm'H$-size protocol.

  ➡**Performers** require $L \geq k$ or $mrH = \Omega(N/k^6)$ to solve hop_k. (Similar results for all kernel-based models and Longformer)



Big idea

- Separation between transformers and alternative models:
    - Transformers are highly parallelizable and can solve tasks like connectivity and multi-hop induction heads with log depth.
    - Other models are not highly parallelizable and must compress their memory of the problem in each computational step, which makes these tasks hard.

Extensions at Google Research

Follow-up projects

“Attend like a graph”
(Bahar Fatemi, Bryan Perozzi, Jonathan Halcrow, Vahab Mirrokni)

- **Context:** Recent “Talk Like a Graph” and “Let Your Graph Do the Talking” papers from OMEGA team benchmarked graph reasoning tasks with trained LLMs.

Prompt
G describes a graph among nodes 0, 1, 2, 3, 4, 5, 6, 7, and 8.
In this graph:
Node 0 is connected to nodes 2 and 3.
Node 1 is connected to nodes 2 and 8.
…
Question: What is the degree of node 4?

- **Question:** Does this framework for transformers provide insight on graph reasoning?

- **Outcomes:**

  - Experimental results: advantages for vanilla transformers over GNN-based models on tasks with “global structure” (e.g. connectivity).

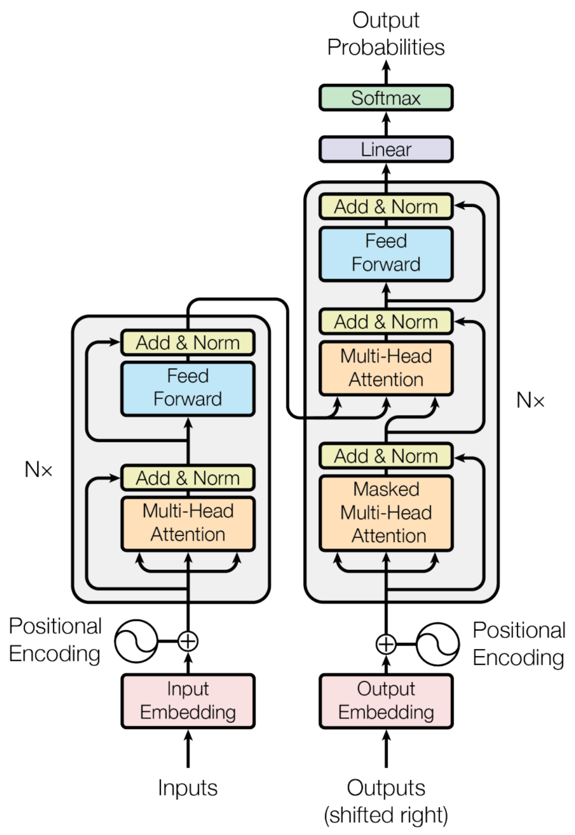
  - Theory results: improvement of MPC reduction and more complete hierarchy of graph reasoning representational results (e.g. triangle counting and shortest path).

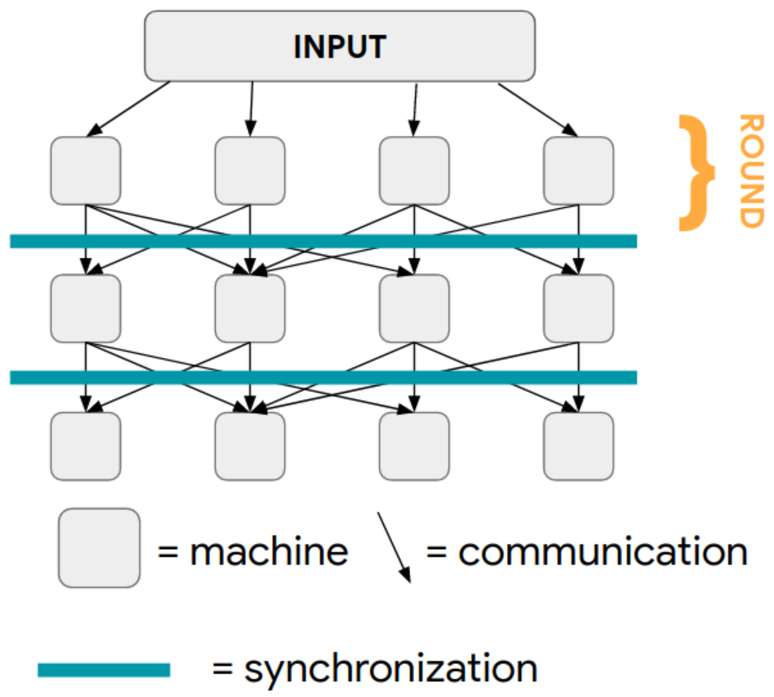
Follow-up projects

Unconditional depth lower bounds (Jieming Mao, Jon Schneider, Vahab Mirrokni)

- **Context:** No existing unconditional negative results for $\omega(1)$-depth transformers.
- **Question:** Is there some task for which $\Theta(\log N)$-depth is necessary and sufficient?
- **Outcomes:**
  - Designed "acausal k-hop induction heads" task likely requires depth $\Omega(k)$; information theoretic proof draft that furthers connection to pointer chasing.
  - Empirical validation of hardness of this task vs standard hop_k.

Our results





Theorem 1: Transformers simulate MPC protocols.

Log-depth Transformers can solve connected components

Theorem 2: MPC protocols simulate transformers.

Transformers require log-depth to solve connected components under 1-cycle vs 2-cycle conjecture.

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

Thank you