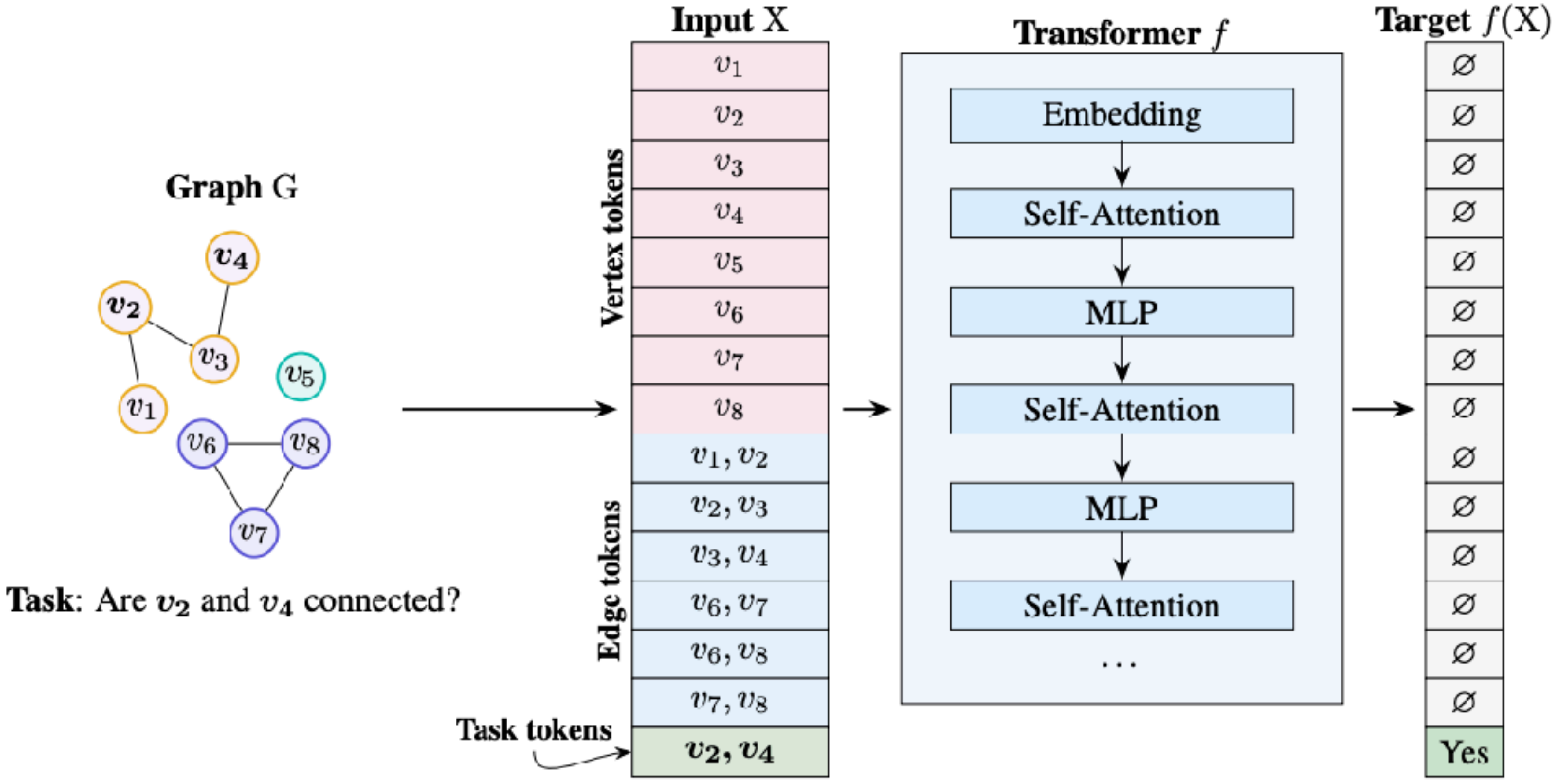
Transformers and graph algorithms

Clayton Sanford

July 12, 2024



Collaboration with Bahare Fatemi, Ethan Hall, Anton Tsitsulin, Mehran Kazemi, Jonathan Halcrow, Bryan Perozzi, Vahab Mirrokni

Google Research

Motivating questions

- Algorithmic powers and limitations of transformer models?
- Transformers as general-purpose neural networks (in comparison to LSTMs, CNNs, GNNs)?

➡Focus on graph algorithmic tasks (e.g. edge existence, connectivity, shortest path) and GNN comparisons.

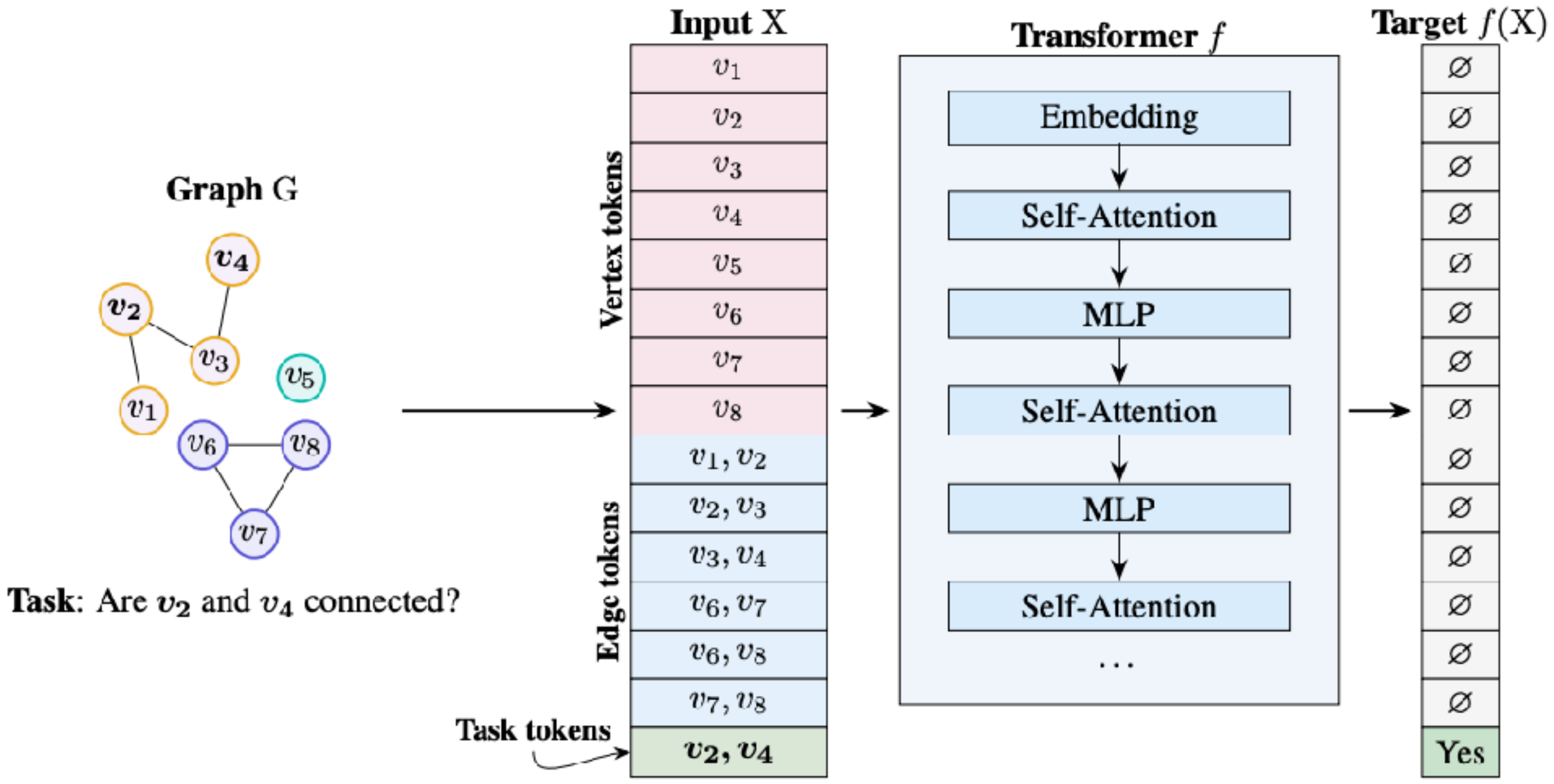
Contributions

Studied graph algorithmic tasks as sequential inputs to “vanilla” transformers.

Theory: representational hierarchy of tasks, contrasts with GNNs.

Empirical: exploratory analysis of learnability of graph tasks:

- Models: transformers vs GNNs.
- Training regimes: trained from scratch, fine-tuning, prompting.

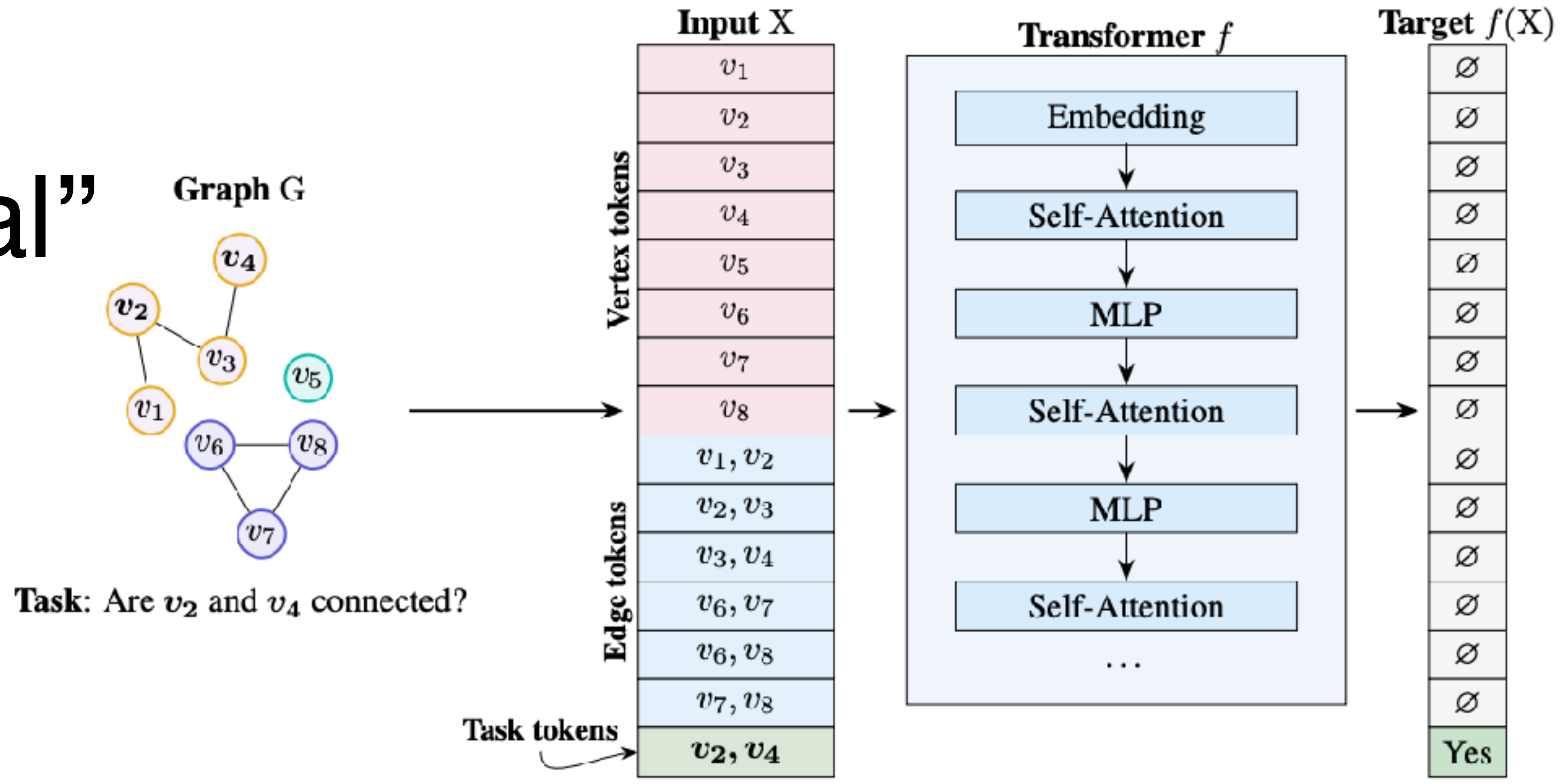


Takeaways

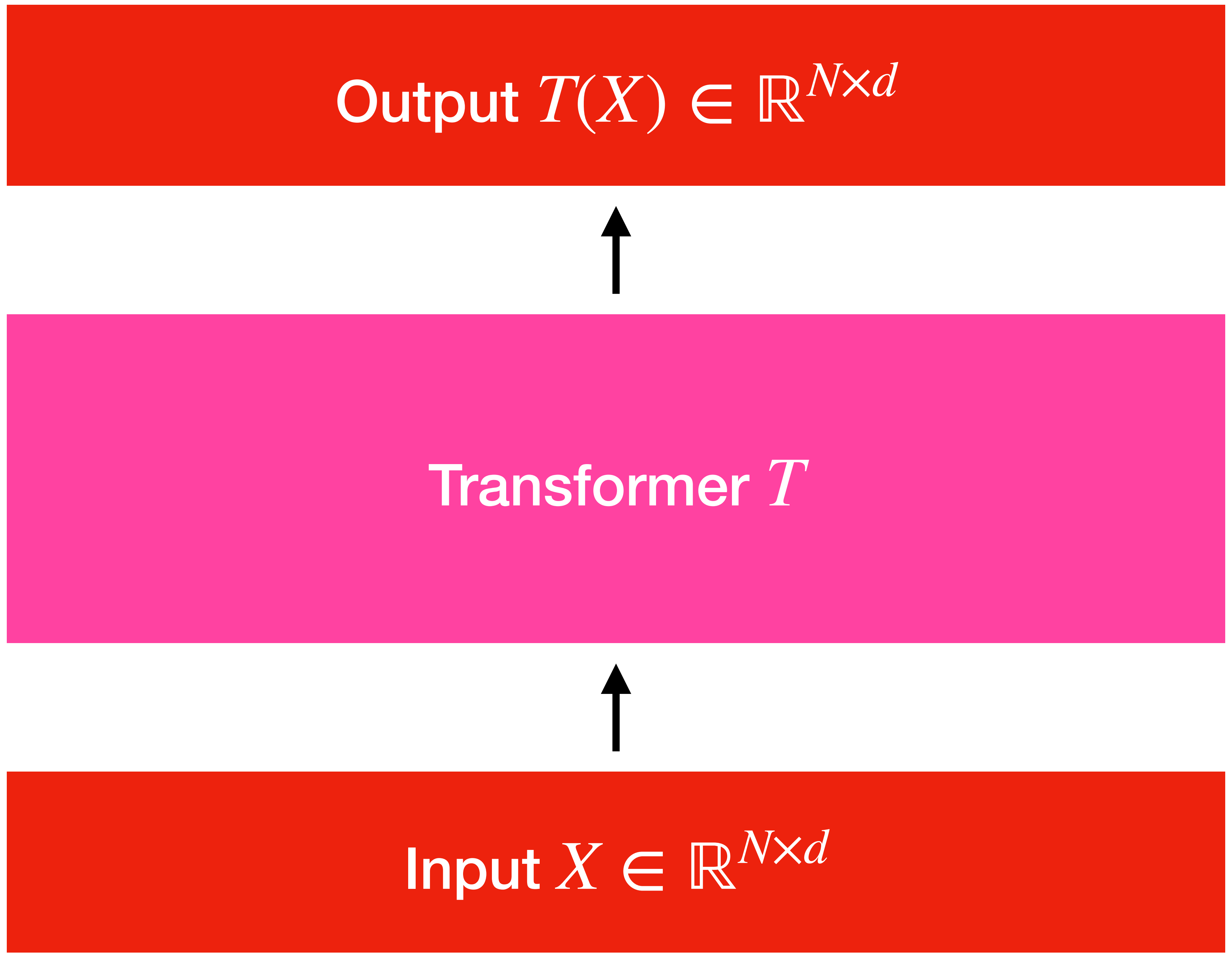
1. GNNs $\gg$ transformers on “local” tasks, like edge existence.

2. Transformers $\gg$ GNNs on “global” and parallelizable tasks, like connectivity.

3. Small transformers (~20M parameters) trained from scratch outperform prompting of LLMs (~10B parameter) on small graphs.



Transformer model

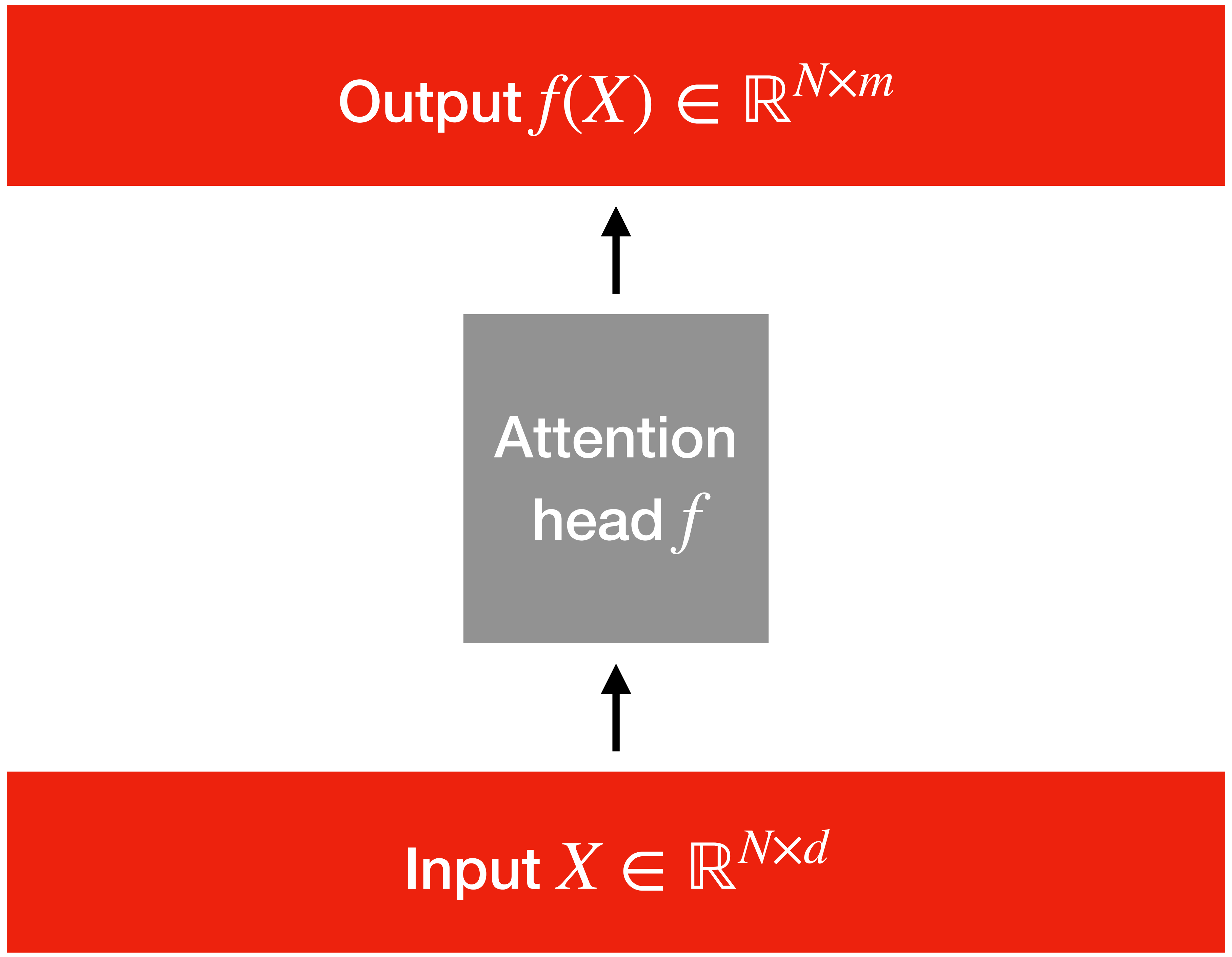


Transformer model

Attention head:

$f(X) = \mathrm{softmax}(XQK^T X^T)XV.$

Parameters: $Q, K, V \in \mathbb{R}^{d \times m}$.

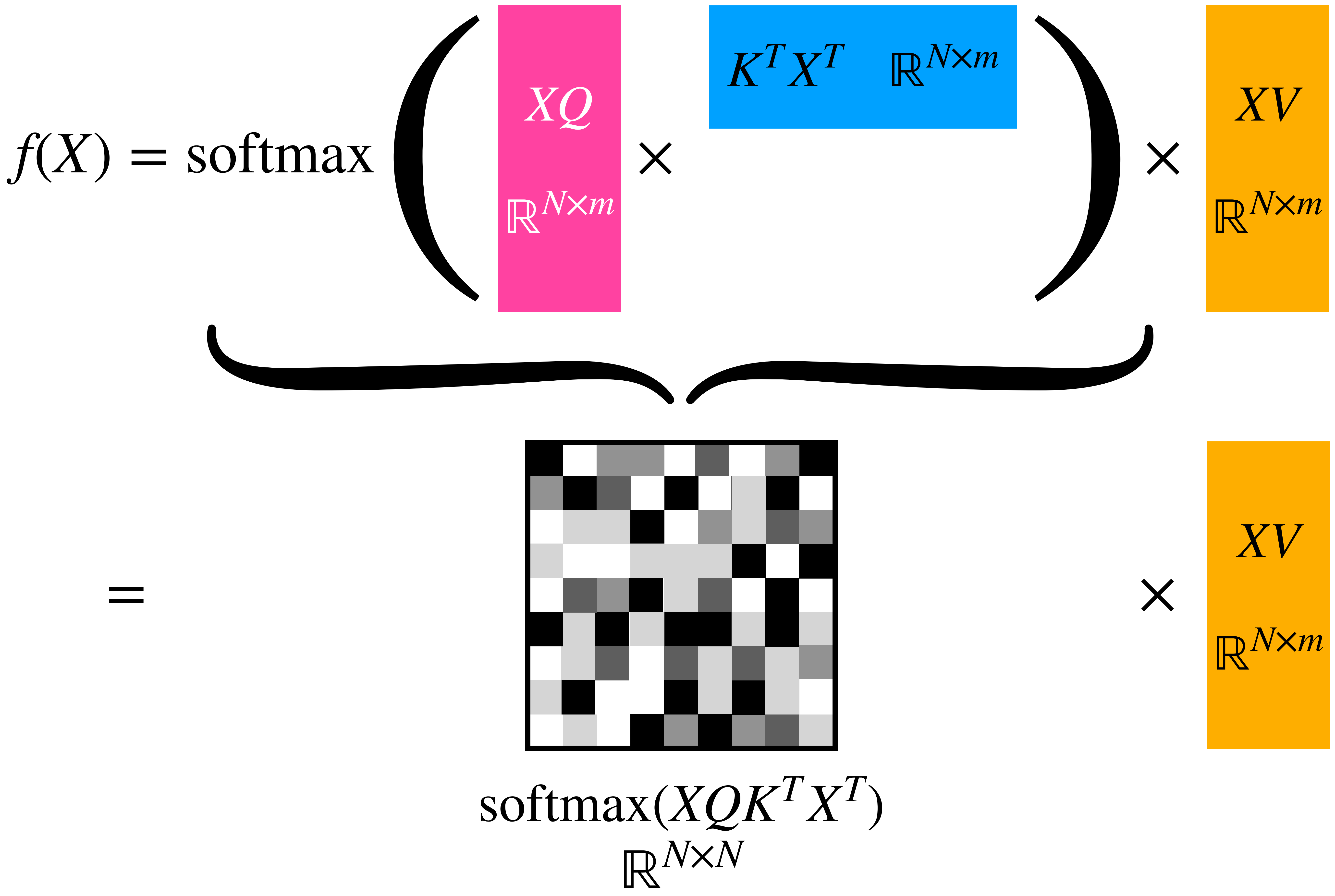


Transformer model

Attention head:

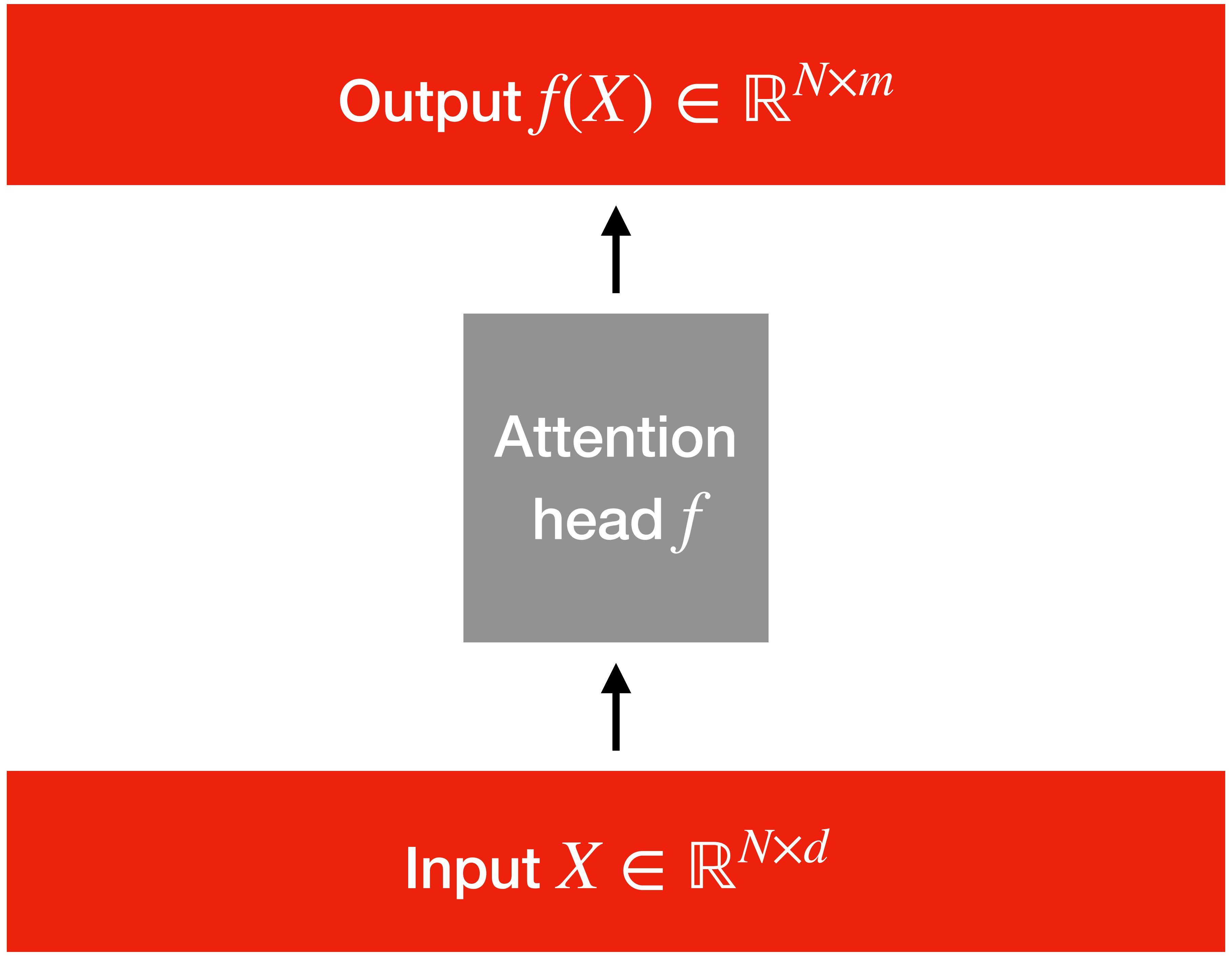
$f(X) = \mathrm{softmax}(XQK^T X^T)XV.$

Parameters: $Q, K, V \in \mathbb{R}^{d \times m}$.



Transformer model

Attention head:
$f(X) = \mathrm{softmax}(XQK^T X^T)XV.$
Parameters: $Q, K, V \in \mathbb{R}^{d \times m}$.



Transformer model

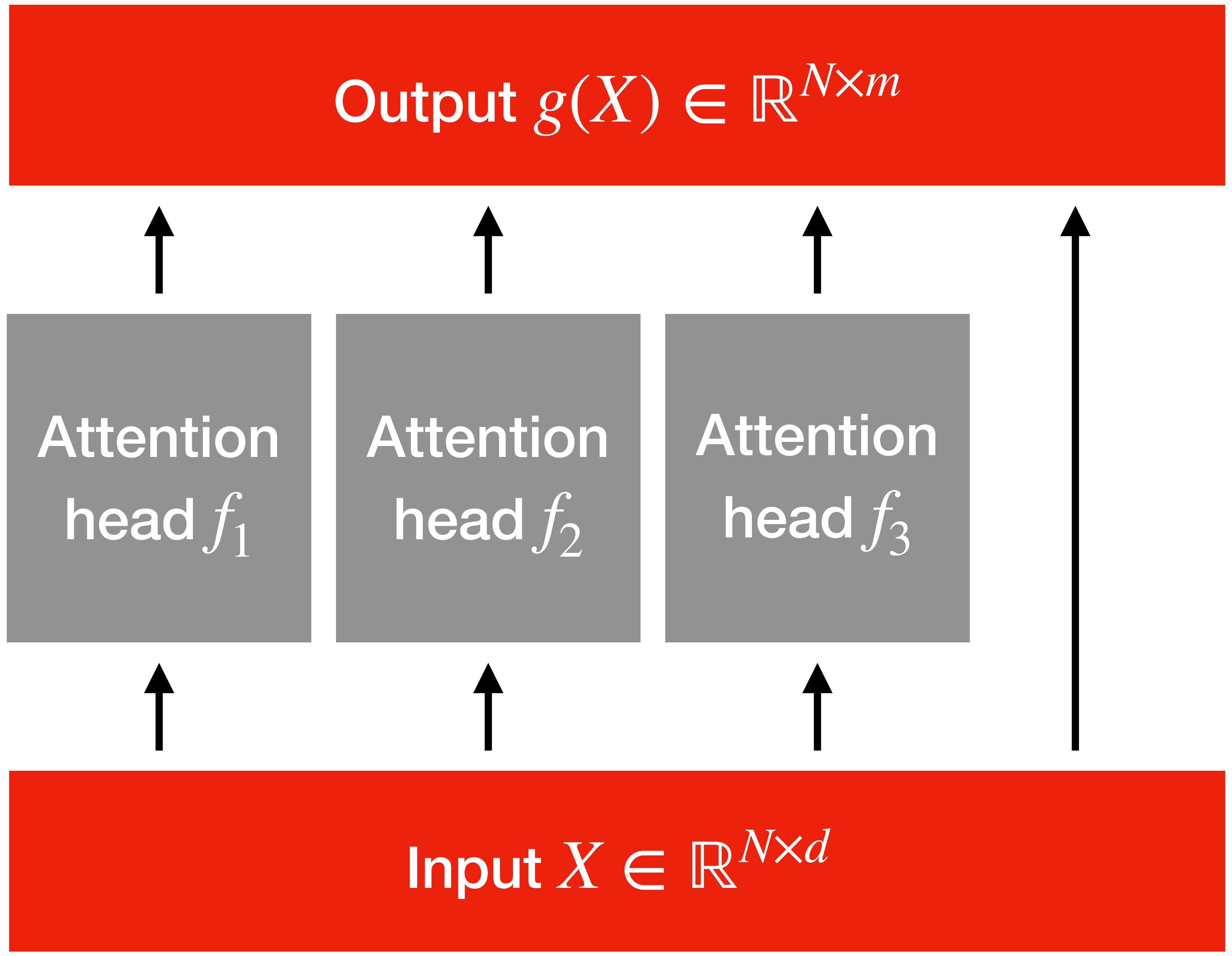
Attention head:

$f(X) = \mathrm{softmax}(XQK^T X^T)XV.$

Parameters: $Q, K, V \in \mathbb{R}^{d \times m}$.

Multi-headed attention:

$$g(X) = X + \sum_{h=1}^{H} f_h(X).$$



Transformer model

Attention head:

$f(X) = \mathrm{softmax}(XQK^TX^T)XV.$

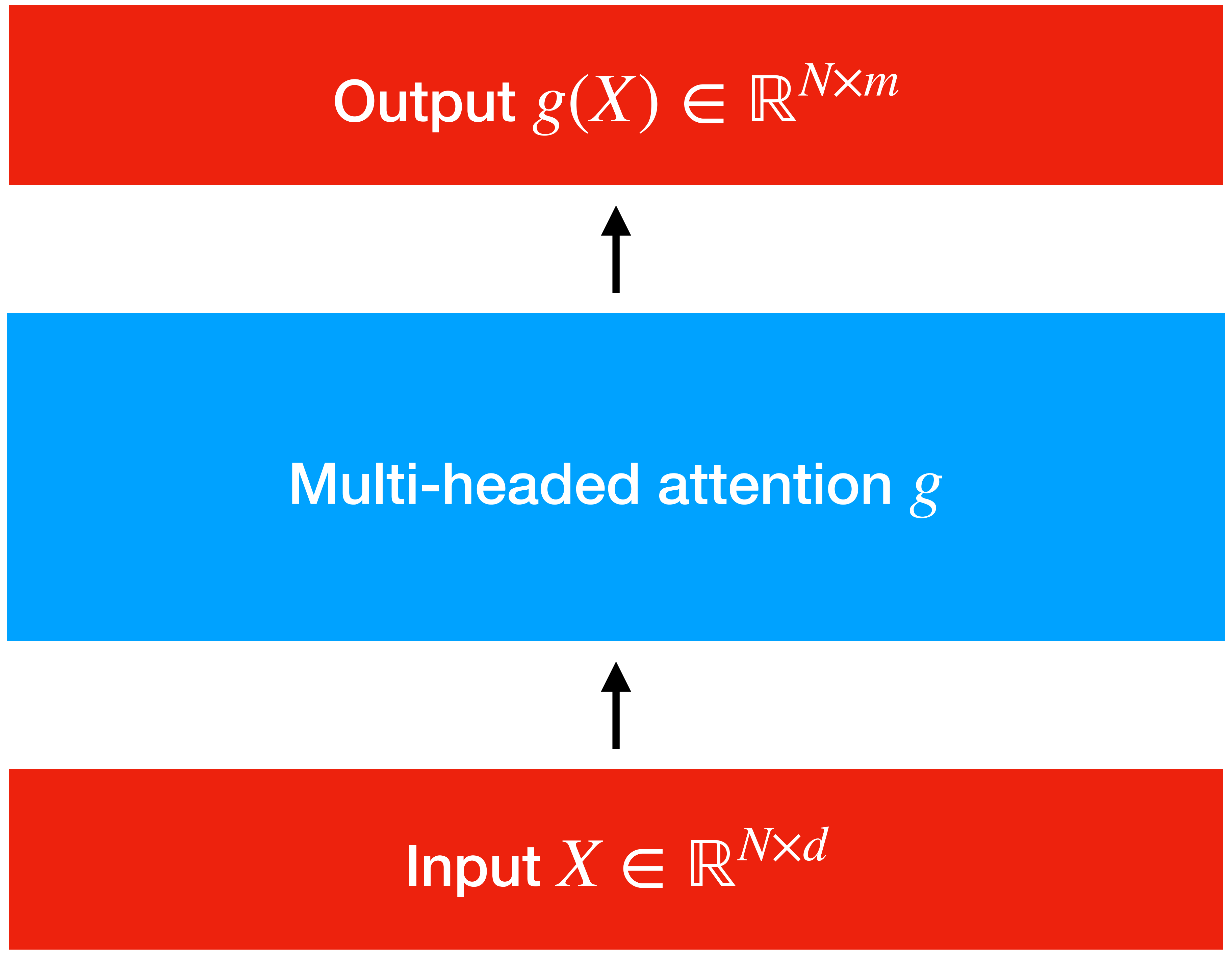
Parameters: $Q, K, V \in \mathbb{R}^{d \times m}$.

Multi-headed attention:

$$g(X) = X + \sum_{h=1}^{H} f_h(X).$$

Element-wise multi-layer perceptron (MLP):

$\phi(X) = (\phi(x_1), \ldots, \phi(x_N)).$



Transformer model

Attention head:

$f(X) = \mathrm{softmax}(XQK^TX^T)XV.$

Parameters: $Q, K, V \in \mathbb{R}^{d \times m}$.

Multi-headed attention:

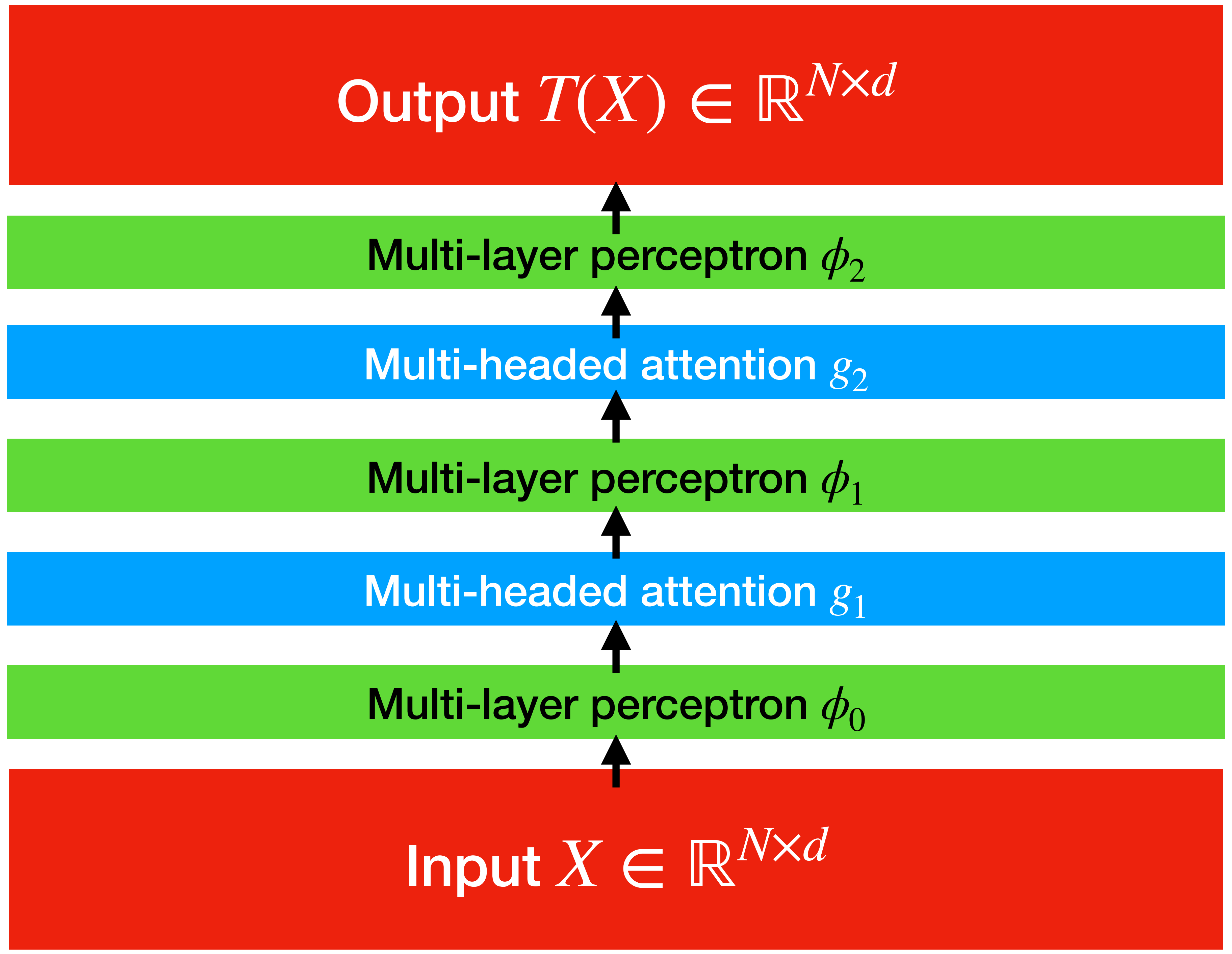
$$g(X) = X + \sum_{h=1}^{H} f_h(X).$$

Element-wise multi-layer perceptron (MLP):

$\phi(X) = (\phi(x_1), \dots, \phi(x_N)).$

Full transformer:

$T(X) = (\phi_L \circ g_L \circ \dots \circ g_1 \circ \phi_0)(X).$



Transformer model

Attention head:
$f(X) = \text{softmax}(XQK^TX^T)XV.$
Parameters: $Q, K, V \in \mathbb{R}^{d\times m}$.

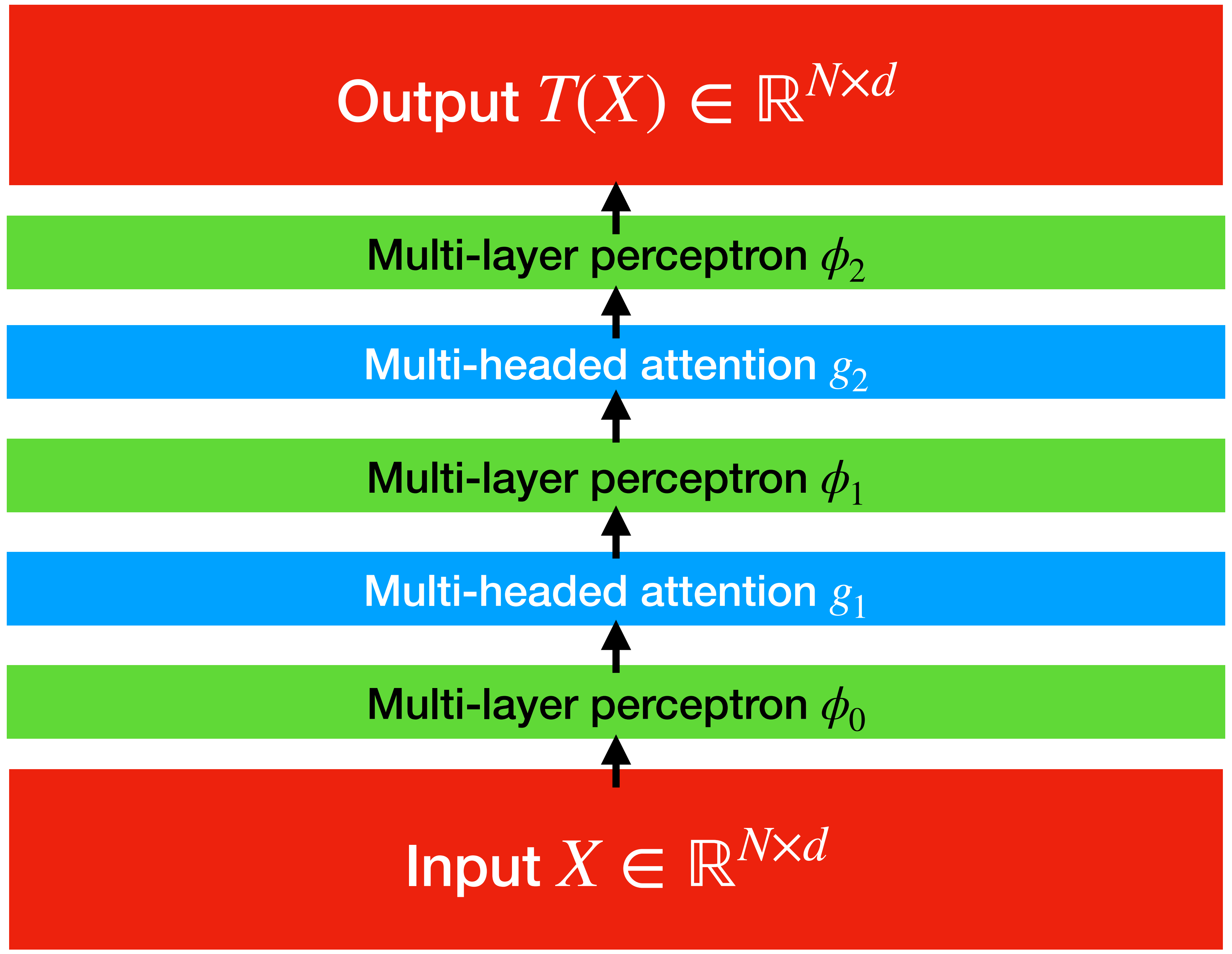
Multi-headed attention:

$$g(X) = X + \sum_{h=1}^{H} f_h(X).$$

Element-wise multi-layer perceptron (MLP):
$\phi(X) = (\phi(x_1), \ldots, \phi(x_N)).$

Full transformer:
$T(X) = (\phi_L \circ g_L \circ \ldots \circ g_1 \circ \phi_0)(X).$

Key assumptions:
$m, H, L \ll N$; arbitrary MLPs ϕ_ℓ.



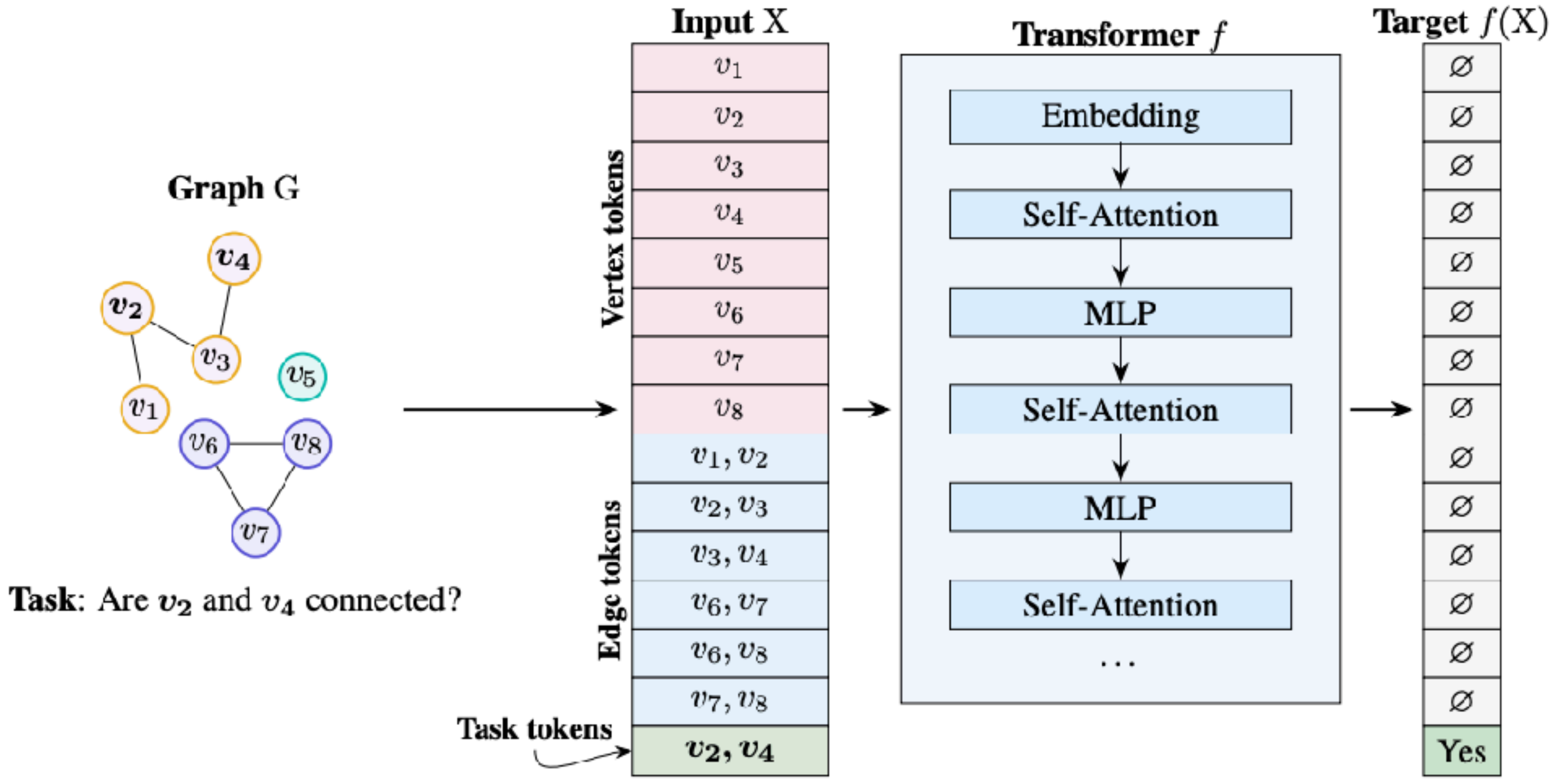
Transformer for graph-based tasks
[Min, et al ’22]

1. **Auxiliary GNNs**: separate transformers and GNNs included in the same model.

2. **Graph positional encoding**: embed graph Laplacians or other info in tokenized positional encoding.

3. **Adjacency-based attention**: soft or hard masking of non-adjacent nodes in positional encoding.

4. **“Pure transformers”**: vanilla transformer models with vertices and edges naively encoded as inputs.



Prior work on transformer capabilities

Inefficient simulation of "serial" algorithms: Turing machines can be simulated by transformers with large depth [Yun, et al '19] or many chain-of-thought tokens [Merrill-Sabharwal '23].

Limitations of constant-depth transformers: Constant-depth transformers can be simulated by TC^0 circuits [MS23].

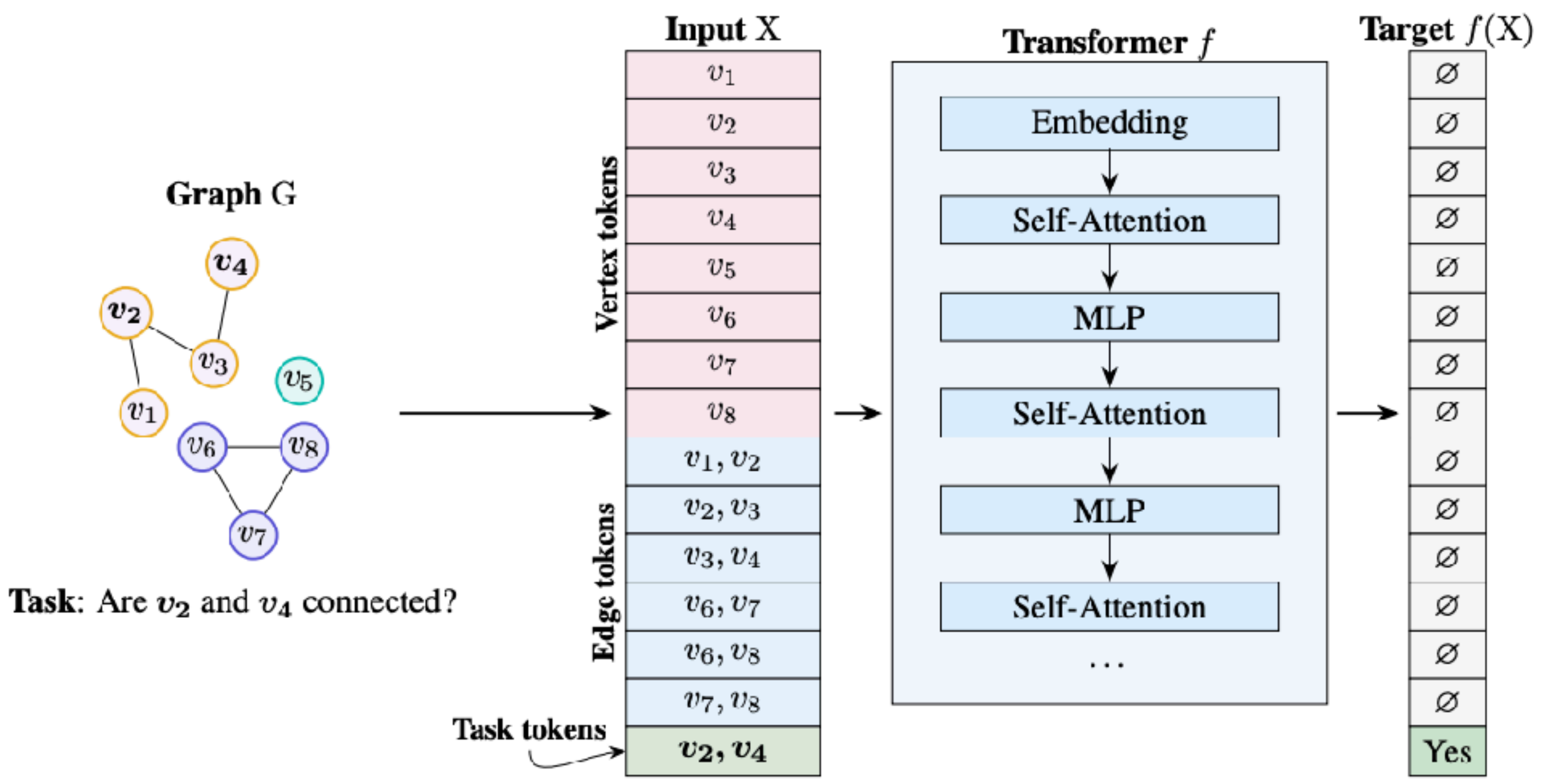
Transformers as communication models: Representational equivalence between transformers and Massively Parallel Computation (MPC) distributed computing model [**S**-Hsu-Telgarsky '23 & '24].

Transformers and graph connectivity

Inefficient simulation of “serial” algorithms:
✅ $\mathrm{poly}(N)$-depth or $\mathrm{poly}(N)$-CoT transformer.

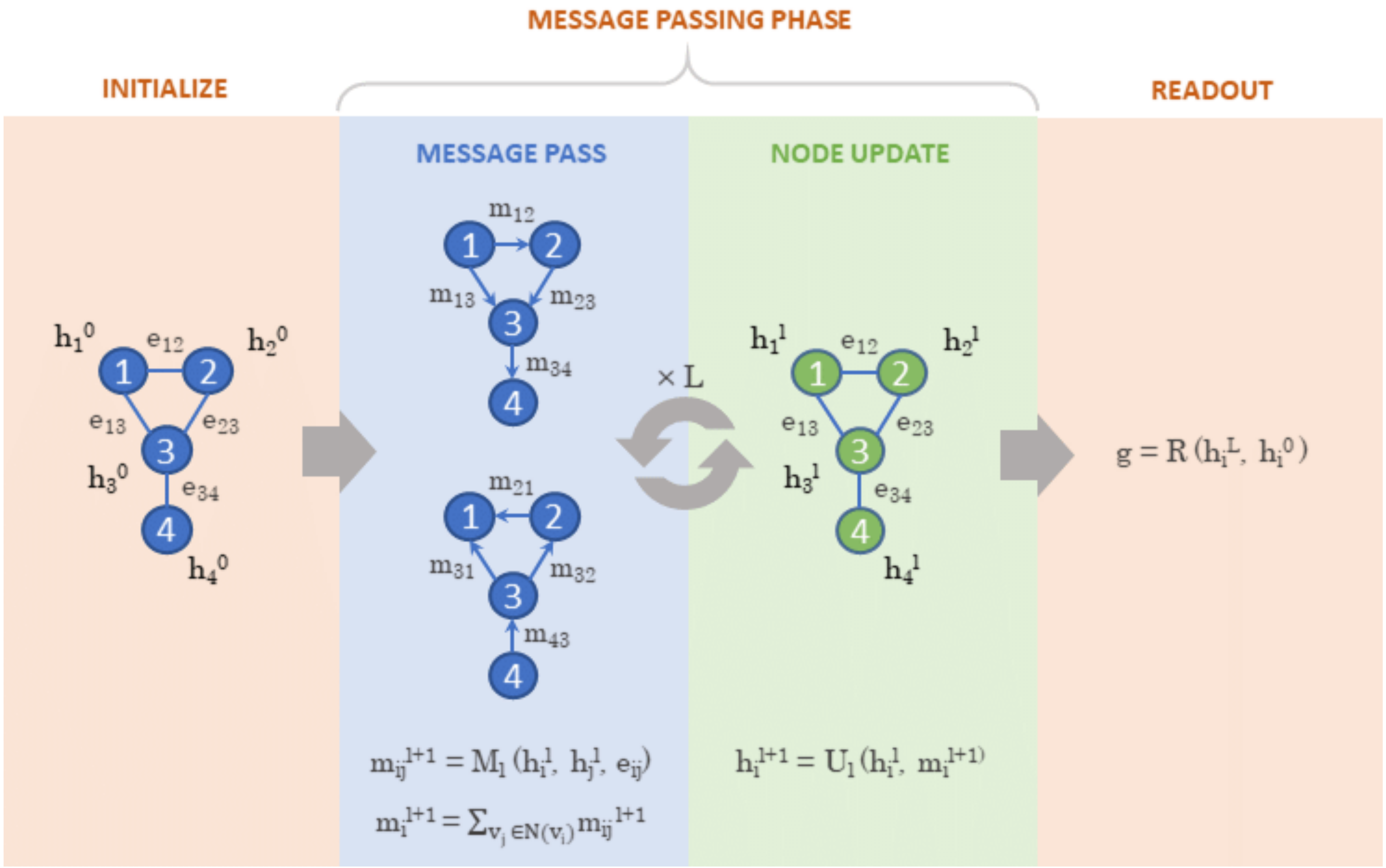
Limitations of constant-depth transformers:
❌ $O(1)$-depth $\mathrm{poly}(N)$-width transformers.

Transformers as communication models:
✅ $\log(N)$-depth $N^{0.1}$-width transformers (optimal depth).



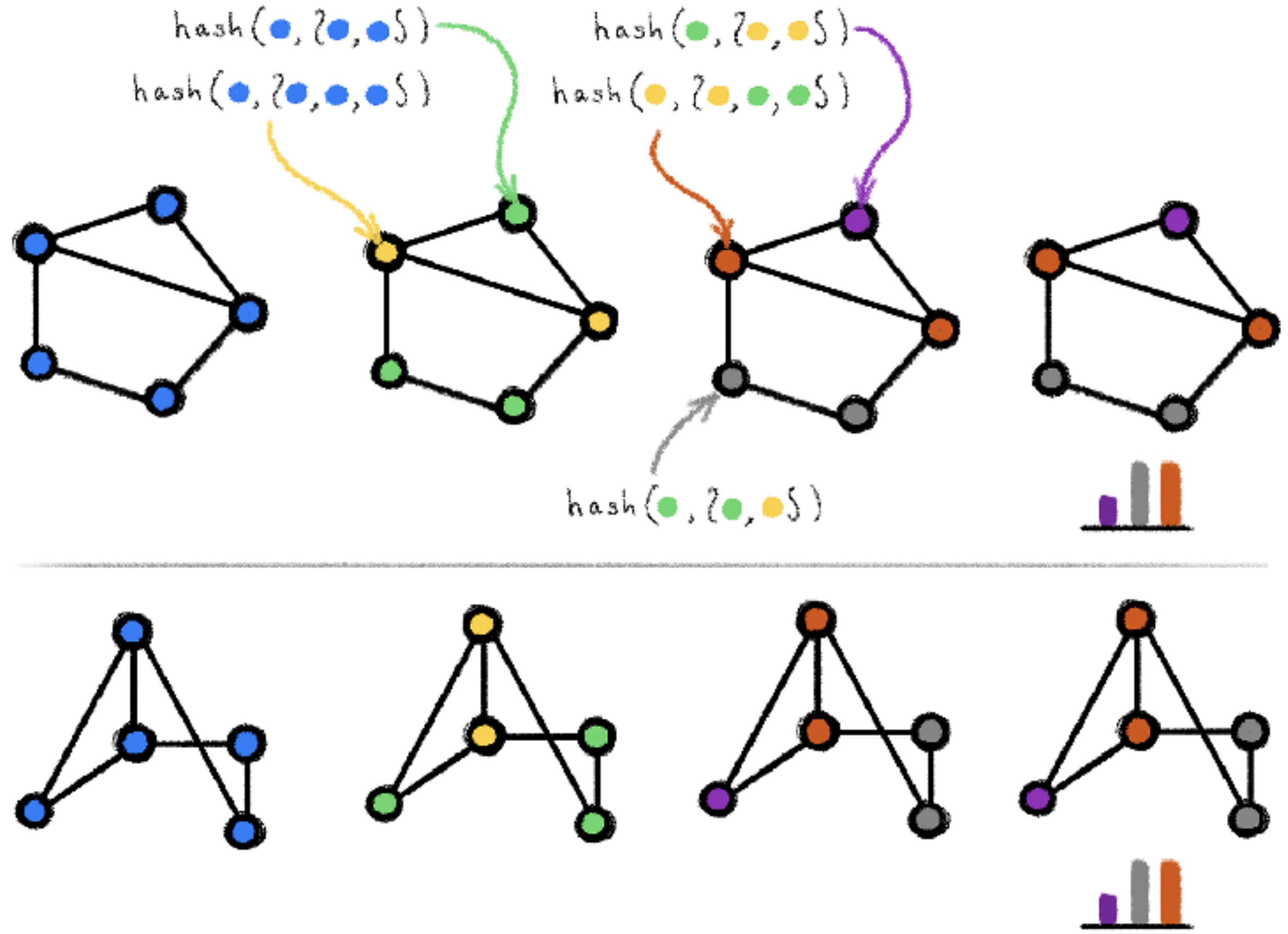
Message-Passing Graph Neural Networks (MPNNs) [Gilmer et al '17]

- Original motivation: chemistry.
- Input graphs restrict sharing of information between adjacent nodes.
- Nodes pass embeddings as "messages" to neighbors and aggregate received messages.



Limitations of GNNs

Weisfeiler-Lehman (WL) isomorphism test:
Featureless GNNs can distinguish non-isomorphic graphs only if distinguishable by WL-test [Xu et al ’18].

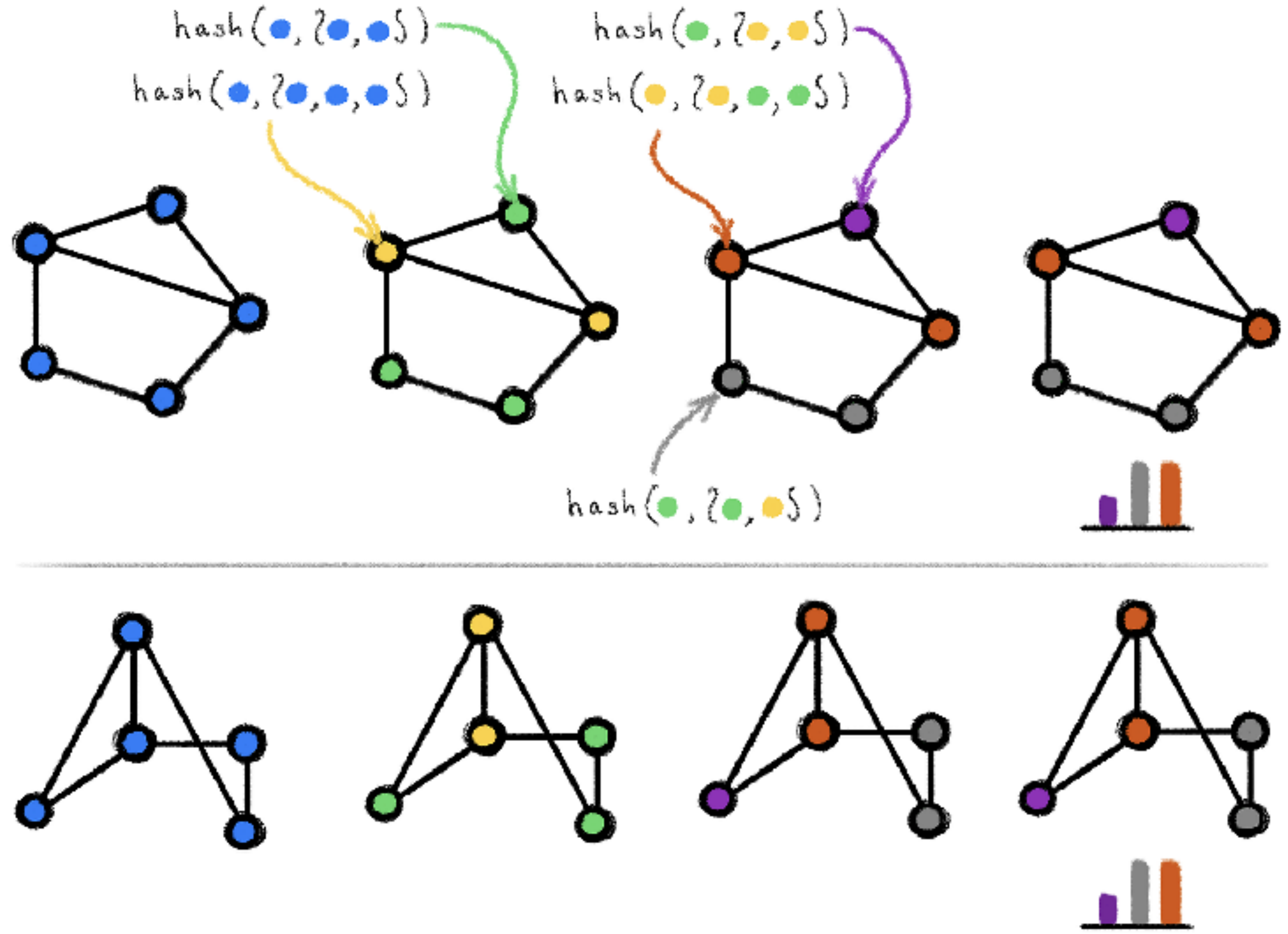


CONGEST: Each GNN layer can be simulated by 1 round of CONGEST distributed computing [Loukas ’19].

GNNs and graph connectivity

Weisfeiler-Lehman (WL) isomorphism test:
❌ featureless GNNs can distinguish between connected and disconnected graphs.

CONGEST:
❌ GNNs solving connectivity with depth L and width m satisfying $L\sqrt{m} = \tilde{O}(N)$.

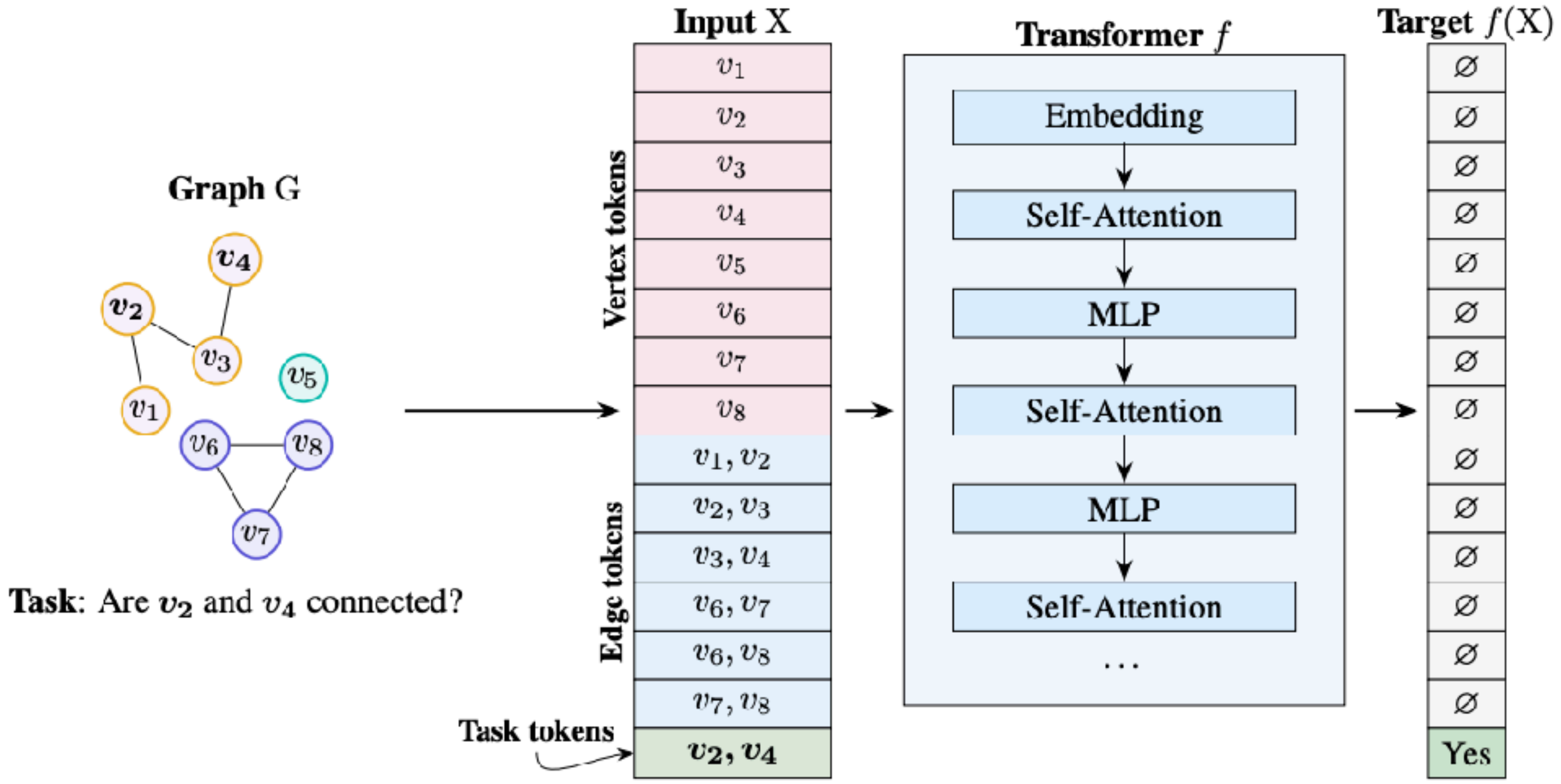


Motivation

Transformers have more parameter-efficient solutions to connectivity than GNNs $(L = O(\log N), m = N^{\epsilon}$ vs $L\sqrt{m} = \Omega(\sqrt{N}))$.

Question 1: Does this apply to other basic graph algorithms tasks?

Question 2: Do transformers outperform GNNs on learnability, not just expressivity?



Theoretical results

Partition of graph algorithmic tasks into transformer parameter-complexity equivalence classes.

- **Retrieval tasks:** node count, edge count, node degree, node existence.
- **Parallelizable tasks**: connectivity, cycle check, minimum spanning forest, # connected components, bipartiteness, planarity.
- **Search tasks**: shortest path, diameter, reachability.

Transformer parameter size regimes:

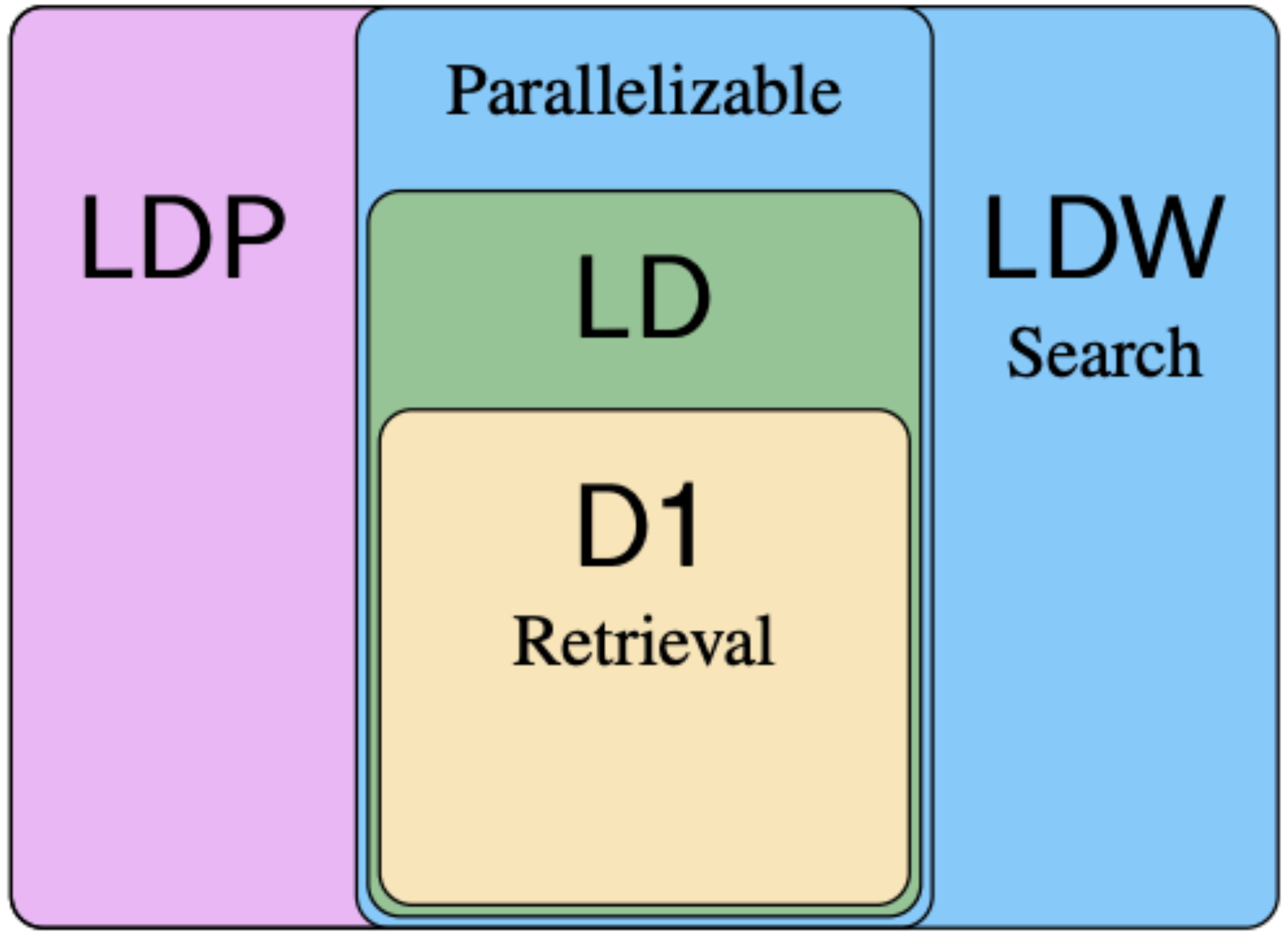
- **Depth 1 (D1)**: depth $L = 1$, width $m = O(N^{\epsilon})$.
- **Log-depth (LD):** $L = O(\log N)$, $m = O(N^{\epsilon})$.
- **Log-depth with blank "pause" tokens (LDP)**: $L = O(\log N)$, $m = O(N^{\epsilon})$, blank tokens $N' = N^{O(1)}$.
- **Log-depth/large width (LDW):** $L = O(\log N)$, $m = O(N^{0.5+\epsilon})$.

Theoretical results

Partition of graph algorithmic tasks into transformer parameter-complexity equivalence classes.

- **Retrieval tasks:** node count, edge count, node degree, node existence.
- **Parallelizable tasks**: connectivity, cycle check, minimum spanning forest, # connected components, bipartiteness, planarity.
- **Search tasks**: shortest path, diameter, reachability.

Task class	Example tasks	Complexity
Retrieval (§3.3) $L = 1$ $m = O(\log N)$	Node count Edge count Edge existence Node degree	D1 D1 D1 D1
Parallelizable (§3.1) $L = O(\log N)$ $m = O(N^{\epsilon})$	Connectivity Cycle check Bipartiteness	LD LDP∩ LDW LDP∩ LDW
Search (§3.2) $L = O(\log N)$ $m = O(N^{1/2+\epsilon})$	Shortest path Diameter	LDW LDW



Depth-1 theoretical results

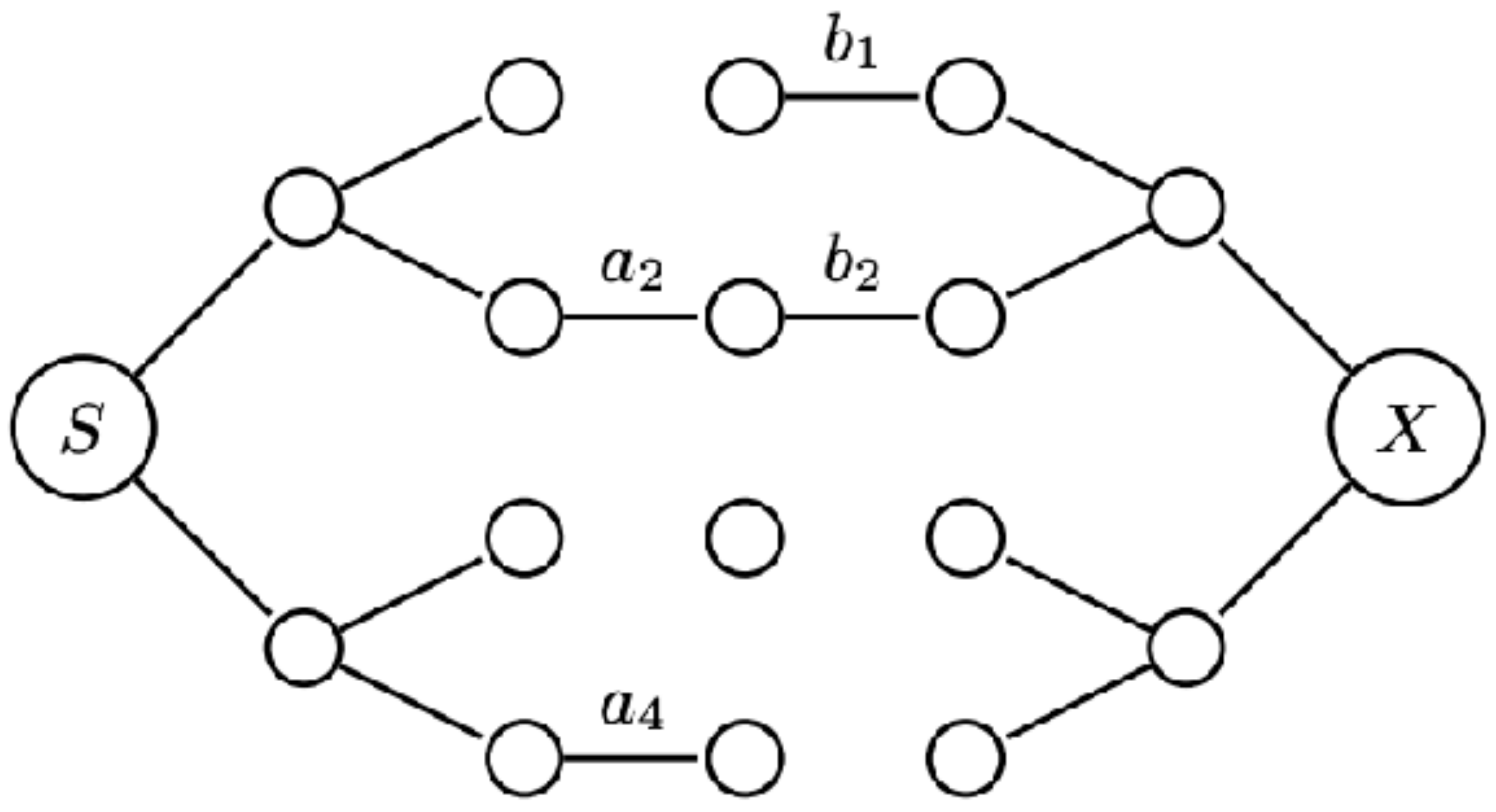
Positive results: There exist D1 transformers ($L = 1, m = O(N^\epsilon)$) that solve all retrieval tasks (node count, edge count, node degree, edge existence).

- Construction depends on sinusoidal embedding of each vertex.

Depth-1 theoretical results

Negative results: No D1 transformer can solve graph connectivity (or cycle check or shortest path).

- **Note**: Already known for constant-depth transformers because these tasks cannot be computed by TC^0 circuits [MS'23].
- Consequence of Alice/Bob communication complexity reduction [**S**-Hsu-Telgarsky '23].
    - Solution to connectivity implies $O(m)$-bit communication protocol for solving disjointness: $\max_i a_i b_i$.



Log-depth theoretical results

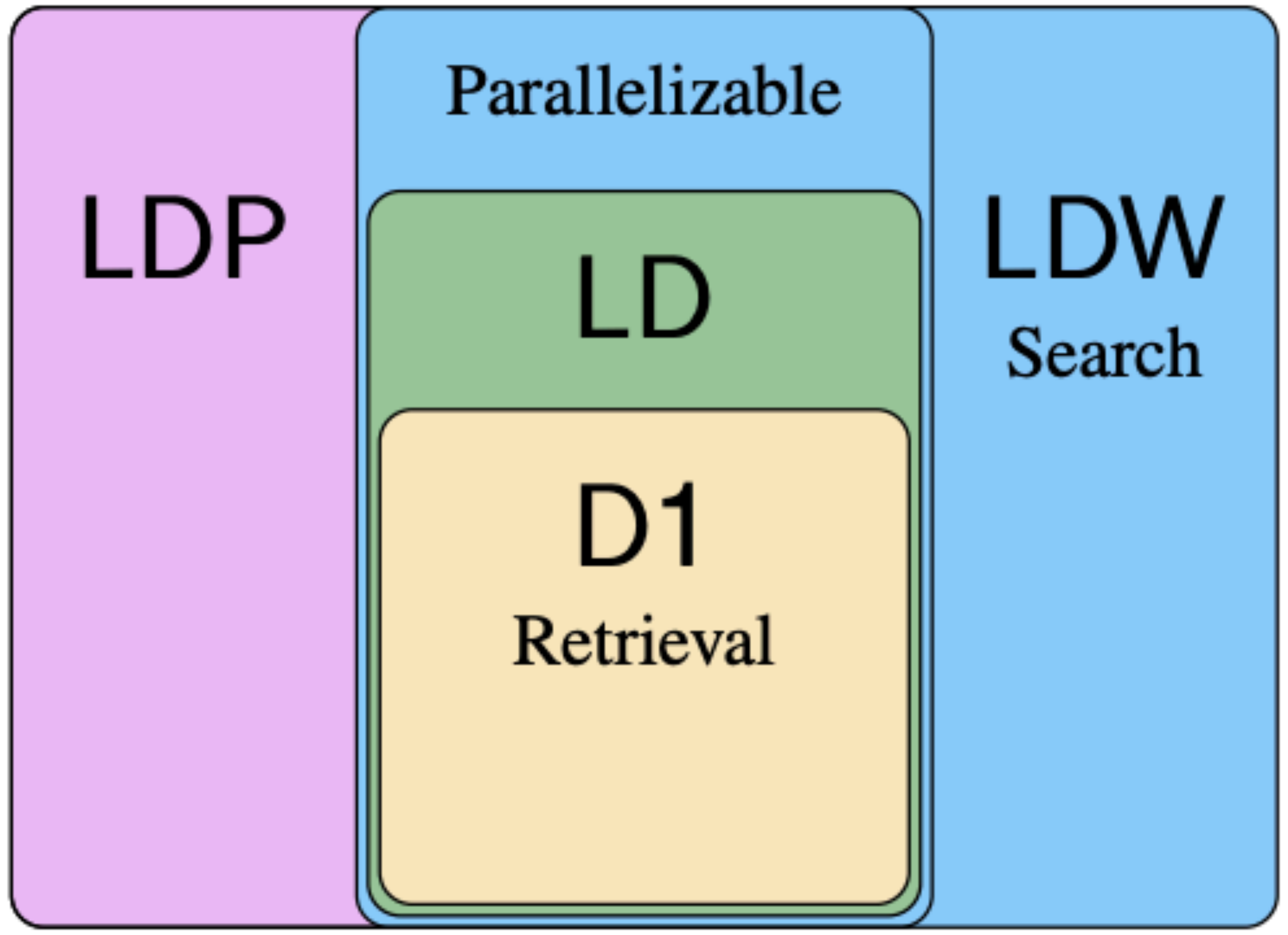
Parallelizable tasks: connectivity, cycle check, minimum spanning forest, # connected components, bipartiteness, planarity.

Parallelizable LDW construction: Transformers of depth $L = O(\log N)$ and width $m = O(N^{0.5+\epsilon})$ can solve any parallelizable task.

Parallelizable LDP construction: Transformers of depth $L = O(\log N)$ and width $m = O(N^{\epsilon})$ with $N' = N^{O(1)}$ blank input tokens can solve any parallelizable task.

Parallelizable log-depth optimality result: All transformers of width $m = O(N^{1-\epsilon})$ and $N' = N^{O(1)}$ blank tokens that solve any parallelizable task have depth $L = \Omega(\log N)$.

Task class	Example tasks	Complexity
Retrieval (§3.3)	Node count	D1
$L = 1$	Edge count	D1
$m = O(\log N)$	Edge existence	D1
	Node degree	D1
Parallelizable (§3.1)	Connectivity	LD
$L = O(\log N)$	Cycle check	LDP∩ LDW
$m = O(N^{\epsilon})$	Bipartiteness	LDP∩ LDW
Search (§3.2)	Shortest path	LDW
$L = O(\log N)$	Diameter	LDW
$m = O(N^{1/2+\epsilon})$		



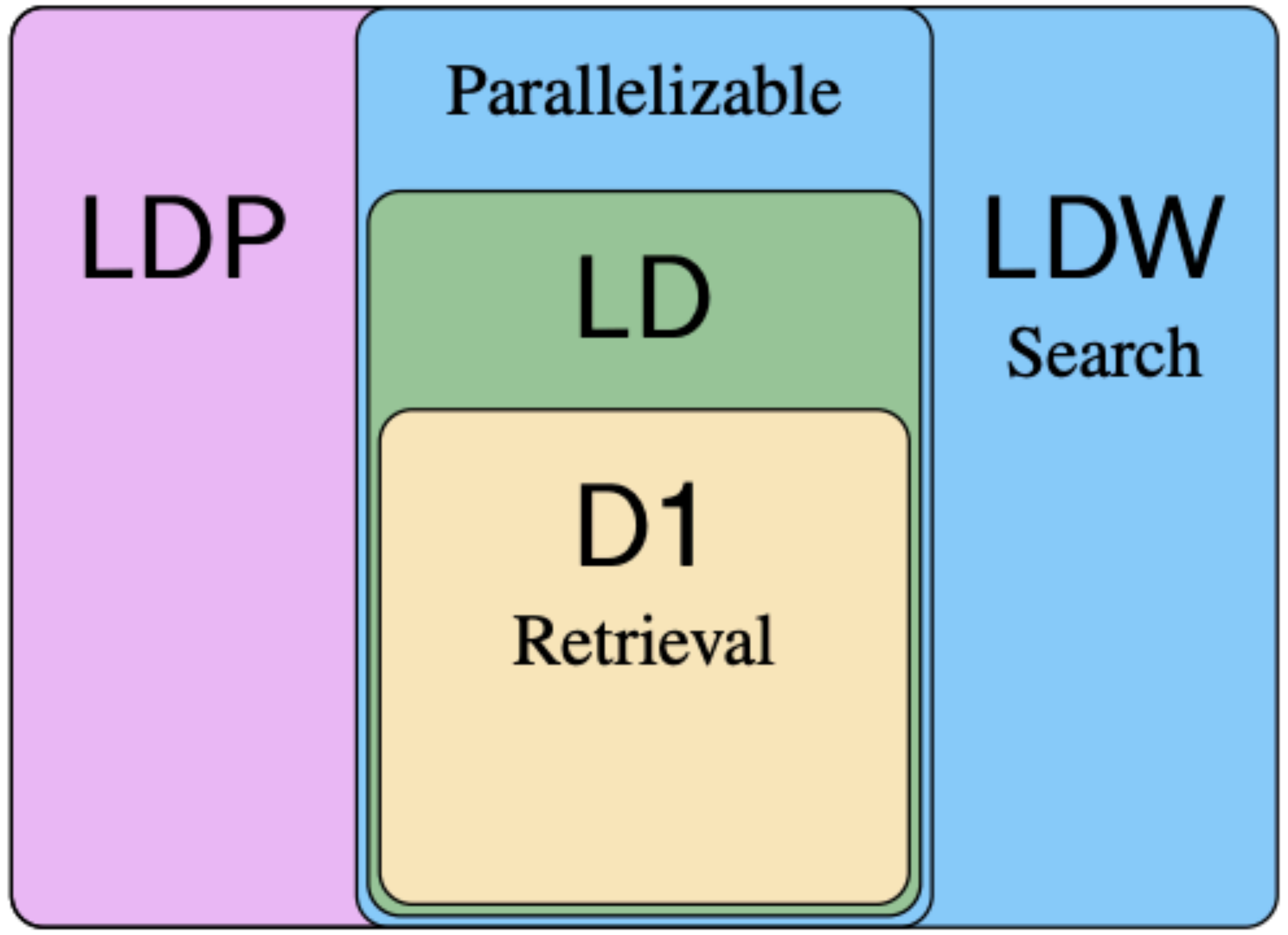
Log-depth theoretical results

Search tasks: shortest path, diameter, reachability.

Search LDW construction: Transformers of depth $L = O(\log N)$ and width $m = O(N^{0.5+\epsilon})$ can solve any search task.

Search depth equivalence: If one search task can be solved by transformers of depth L, width $m = N^{O(1)}$, and $N' = N^{O(1)}$ pause tokens, then all search tasks can be solved with depth $L + O(1)$, width $O(m)$ and $O(N') + N^{O(1)}$ phase tokens.

Task class	Example tasks	Complexity
Retrieval (§3.3) $L = 1$ $m = O(\log N)$	Node count	D1
	Edge count	D1
	Edge existence	D1
	Node degree	D1
Parallelizable (§3.1) $L = O(\log N)$ $m = O(N^{\epsilon})$	Connectivity	LD
	Cycle check	LDP∩ LDW
	Bipartiteness	LDP∩ LDW
Search (§3.2) $L = O(\log N)$ $m = O(N^{1/2+\epsilon})$	Shortest path	LDW
	Diameter	LDW

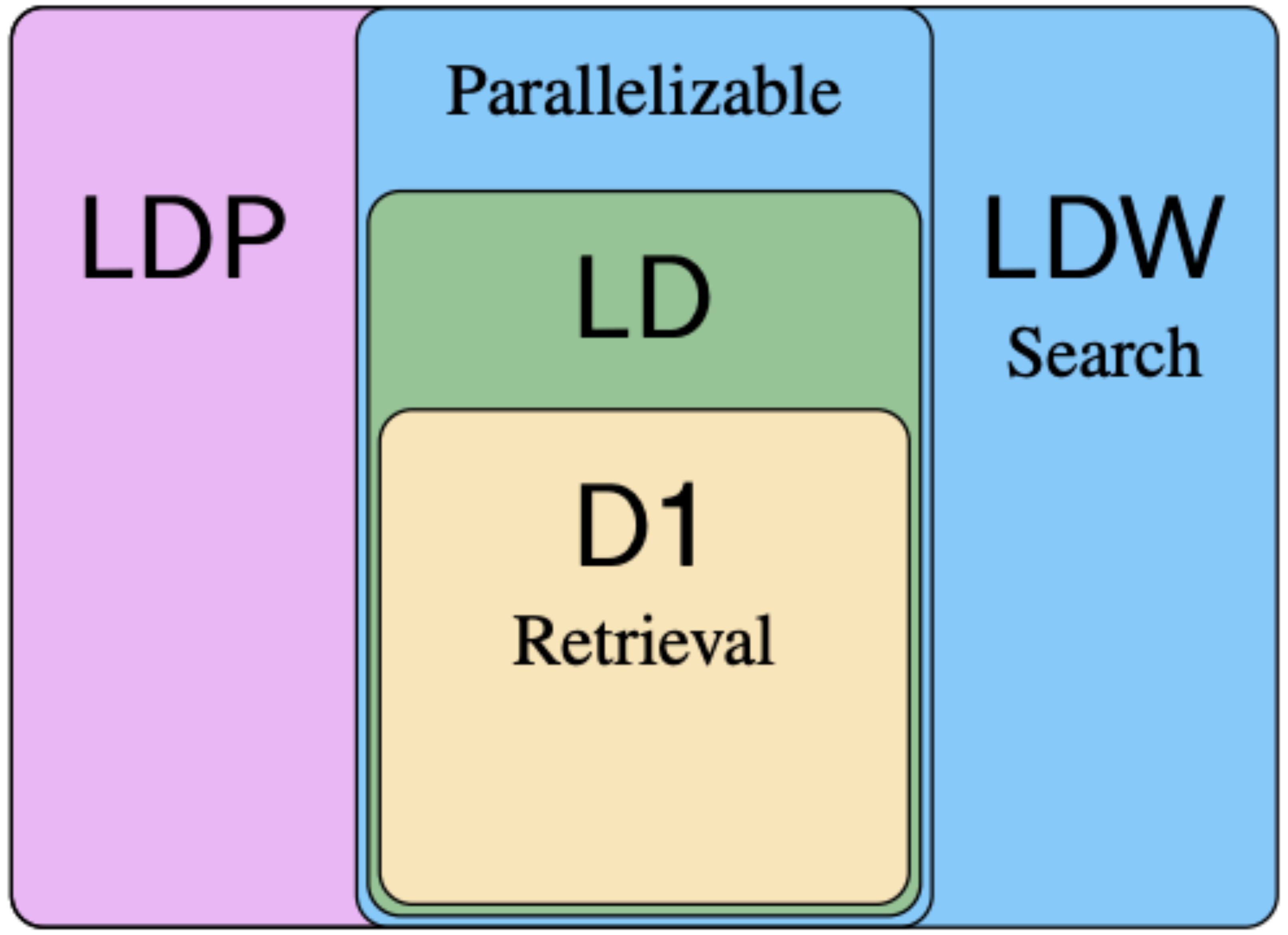


Log-depth proof ideas

Component 1: Bidirectional relationship between transformers and MPC distributed computing model [**S**-Hsu-Telgarsky ’24].

Component 2: Equivalence classes of graph algorithmic tasks in MPC model [Nanongkai-Scquizzato ’22].

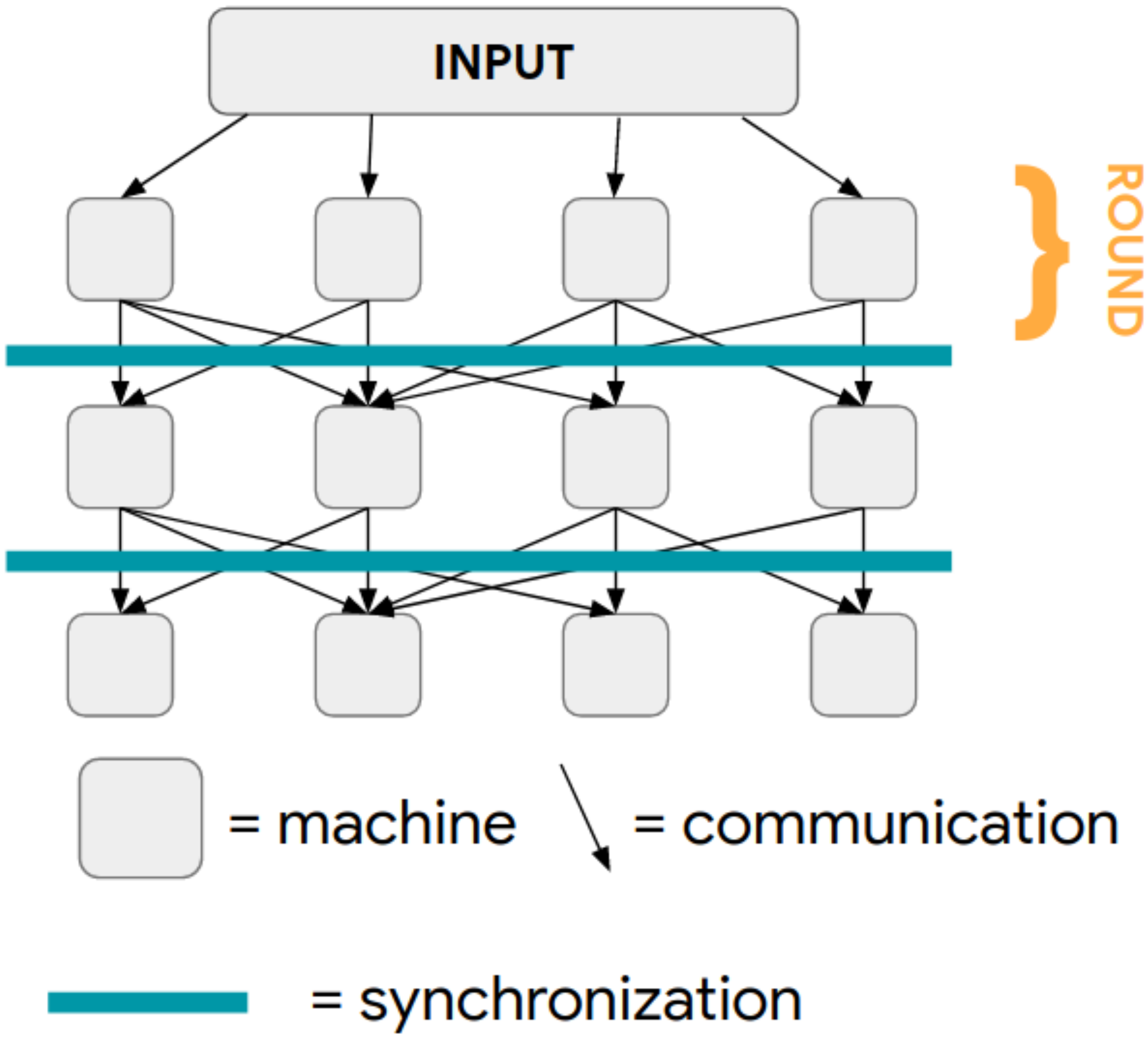
Task class	**Example tasks**	**Complexity**
Retrieval (§3.3) $L = 1$ $m = O(\log N)$	Node count Edge count Edge existence Node degree	D1 D1 D1 D1
Parallelizable (§3.1) $L = O(\log N)$ $m = O(N^\epsilon)$	Connectivity Cycle check Bipartiteness	LD LDP∩ LDW LDP∩ LDW
Search (§3.2) $L = O(\log N)$ $m = O(N^{1/2+\epsilon})$	Shortest path Diameter	LDW LDW



Massively Parallel Computation (MPC)

Computational model of MapReduce [Karloff et al, '10]

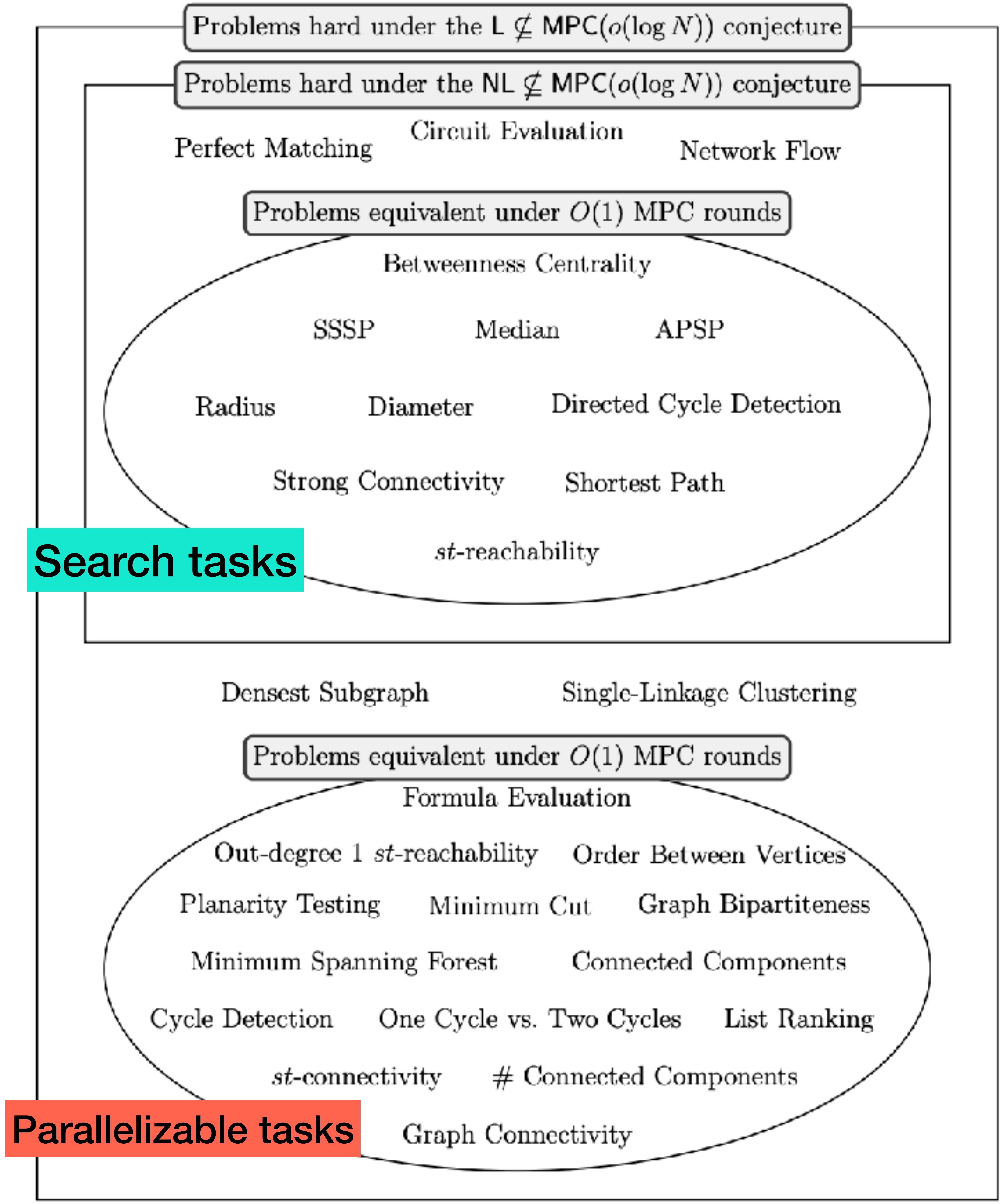
- Input divided among $q = O(N^\delta)$ machines with local memory s ($qs = O(N^{1+\gamma})$).
- Round $r = 1, \ldots, R$:
    - Each machine performs computations on local memory.
    - Each machine sends and receives $\leq s$ bits of information.



Component 2: MPC graph equivalence classes

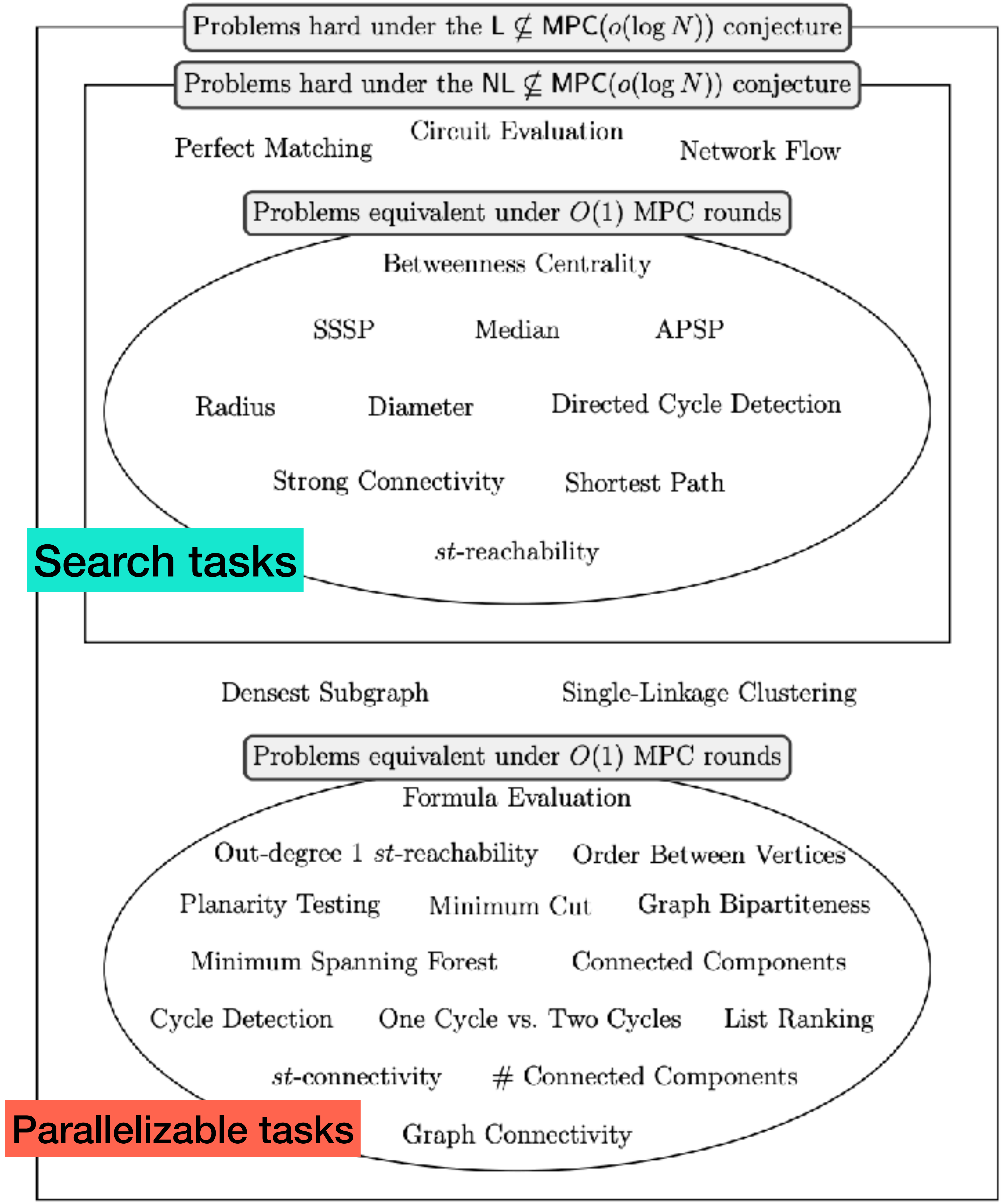
Low memory equivalence:
If a parallelizable task can be solved by an MPC protocol with $s = O(N^{\epsilon})$ local memory, R rounds, and $q = N^{O(1)}$ machines, then all parallelizable tasks can be solved with $O(s)$ local memory, $R + O(1)$ rounds, $q + N^{O(1)}$ machines.

- Positive theorem: Connectivity can be solved with $s = O(N^{\epsilon})$, $R = O(\log N)$, $qs = O(N^{1+\epsilon})$.
- Negative conjecture: connectivity requires $R = \Omega(\log N)$ if $s = O(N^{1-\epsilon})$, $qs = N^{O(1)}$.



Component 2: MPC graph equivalence classes

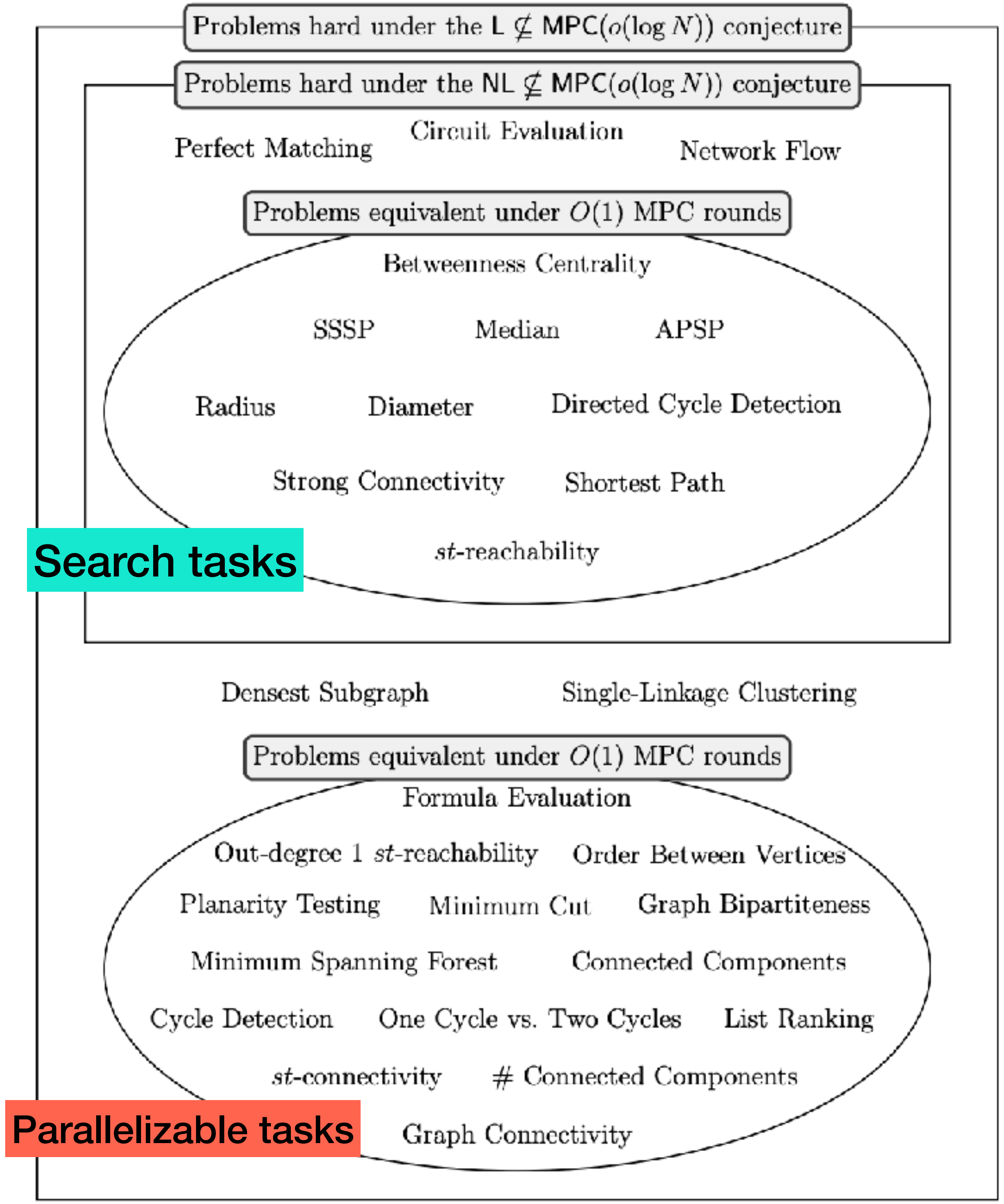
Low memory equivalence:
If a search task can be solved by an MPC protocol with $s = O(N^{\epsilon})$ local memory, R rounds, and $q = N^{O(1)}$ machines, then all search tasks can be solved with $O(s)$ local memory, $R + O(1)$ rounds, $q + N^{O(1)}$ machines.



Component 2: MPC graph equivalence classes

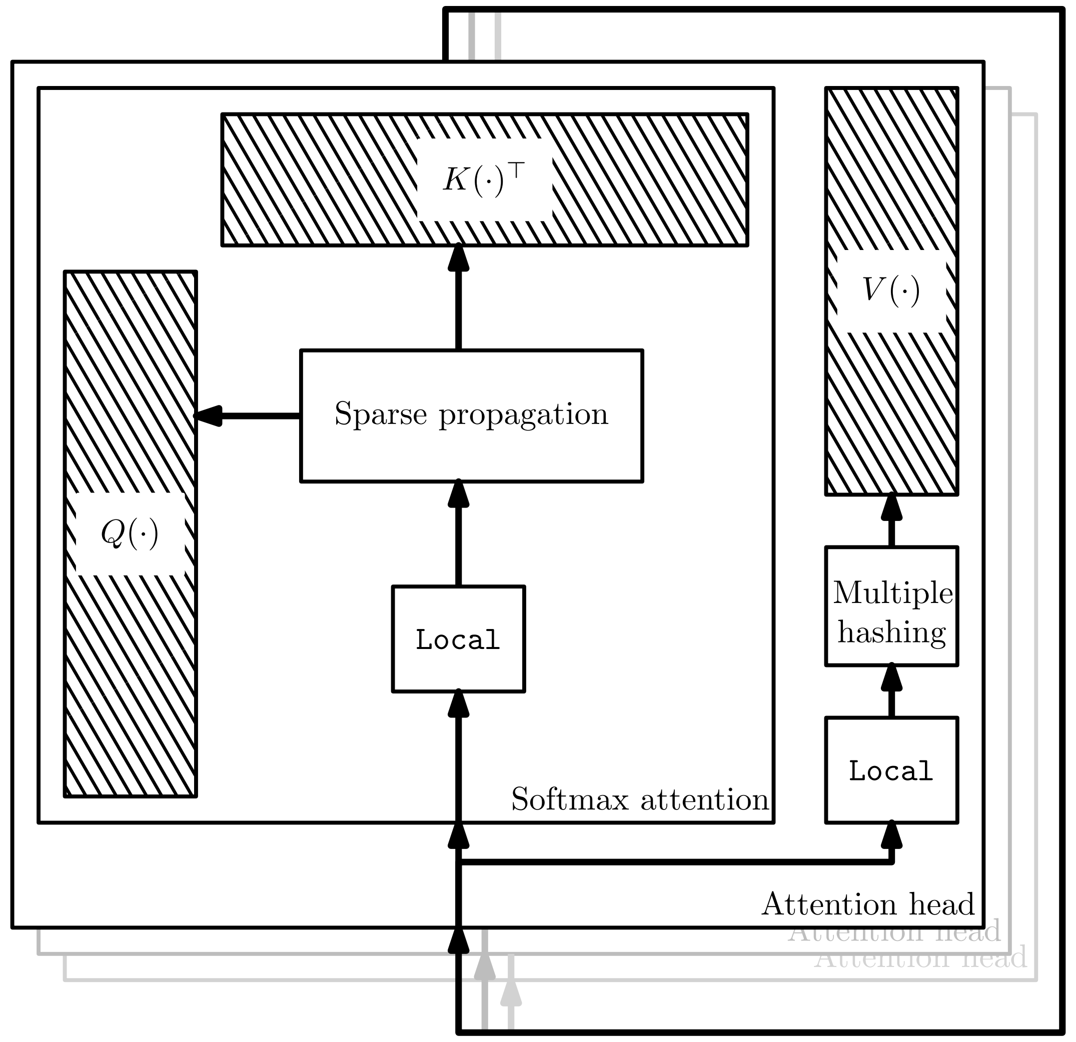
High memory capability:
All NC¹ circuits can be simulated with $s = O(N^{0.5+\epsilon})$ local memory, $q = O(N)$ machines, $L = O(\log N)$ rounds.

- All parallelizable and search tasks belong to L and NL.
- $\mathsf{L} \subseteq \mathsf{NL} \subseteq \mathsf{NC}^1$.

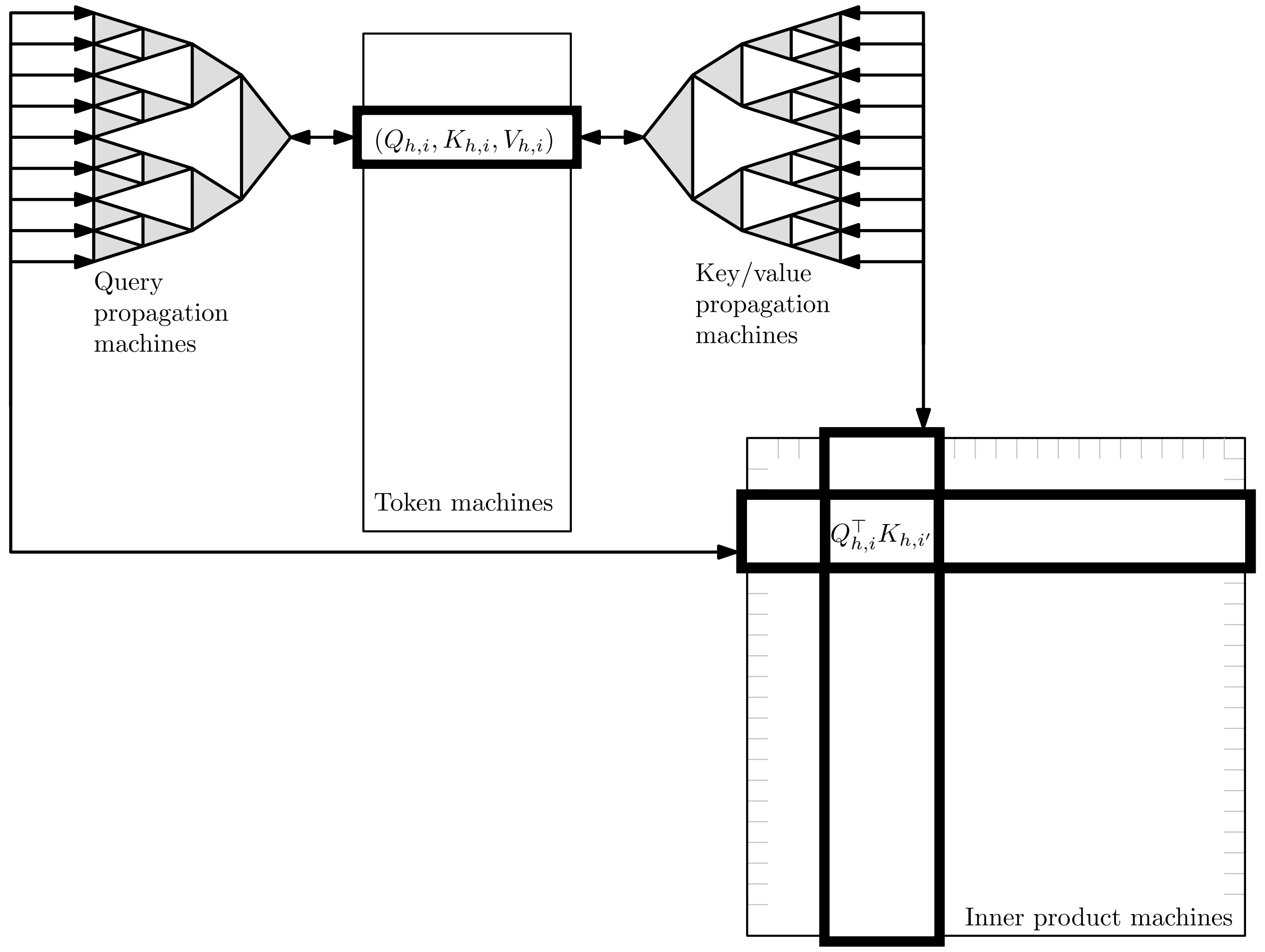


Component 1: Transformer/MPC relationship

Transformers simulate MPC [SHT24]:
R-round MPC protocols with local memory s, # machines q can be simulated by transformers of depth $L = R + 1$, width $m = \tilde{O}(s^4 \log q)$.



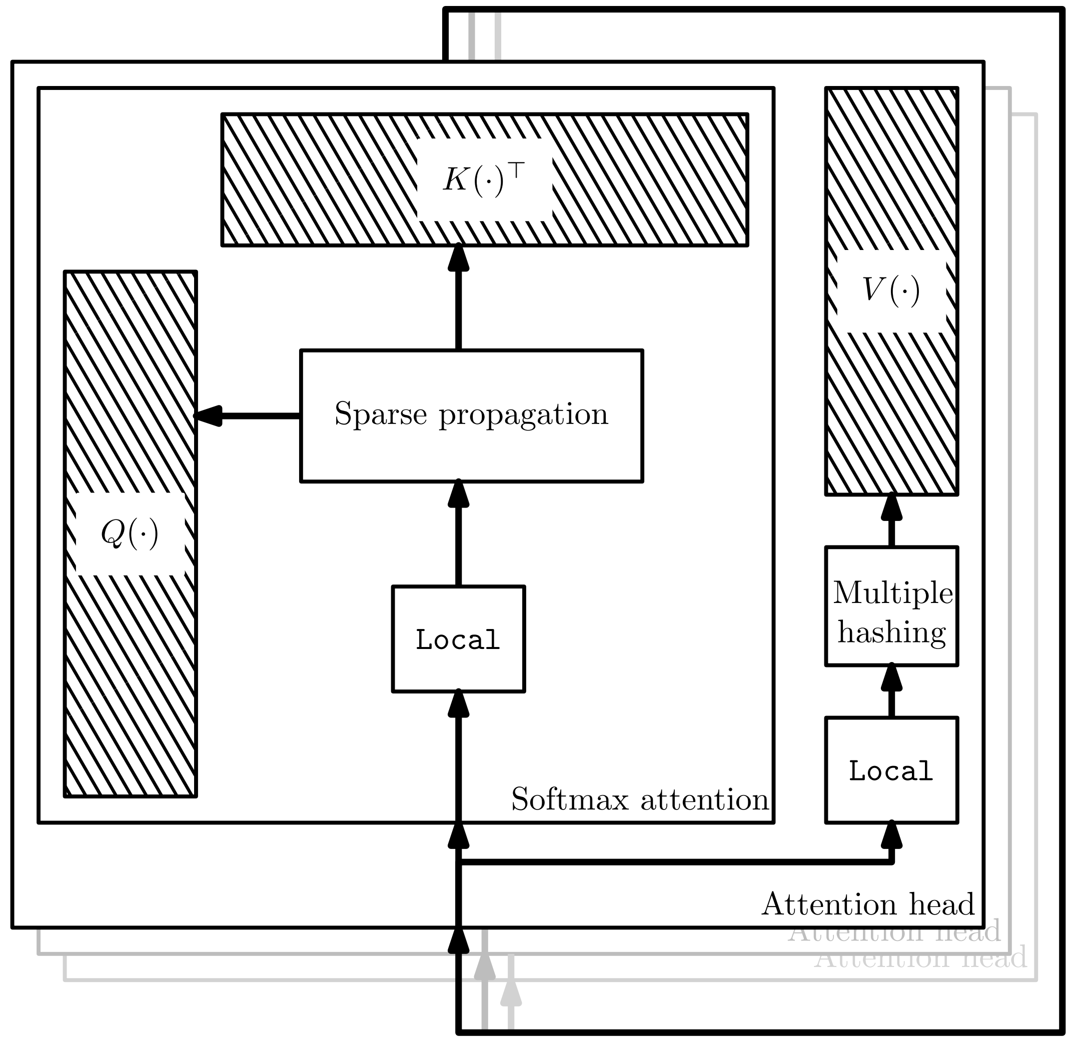
MPC simulates transformers [SHT24]:
Transformers with depth L, width m can be simulated by MPC protocols with $R = O(L)$ rounds, local memory $s = O(m)$, # machines $q = O(N^2)$.

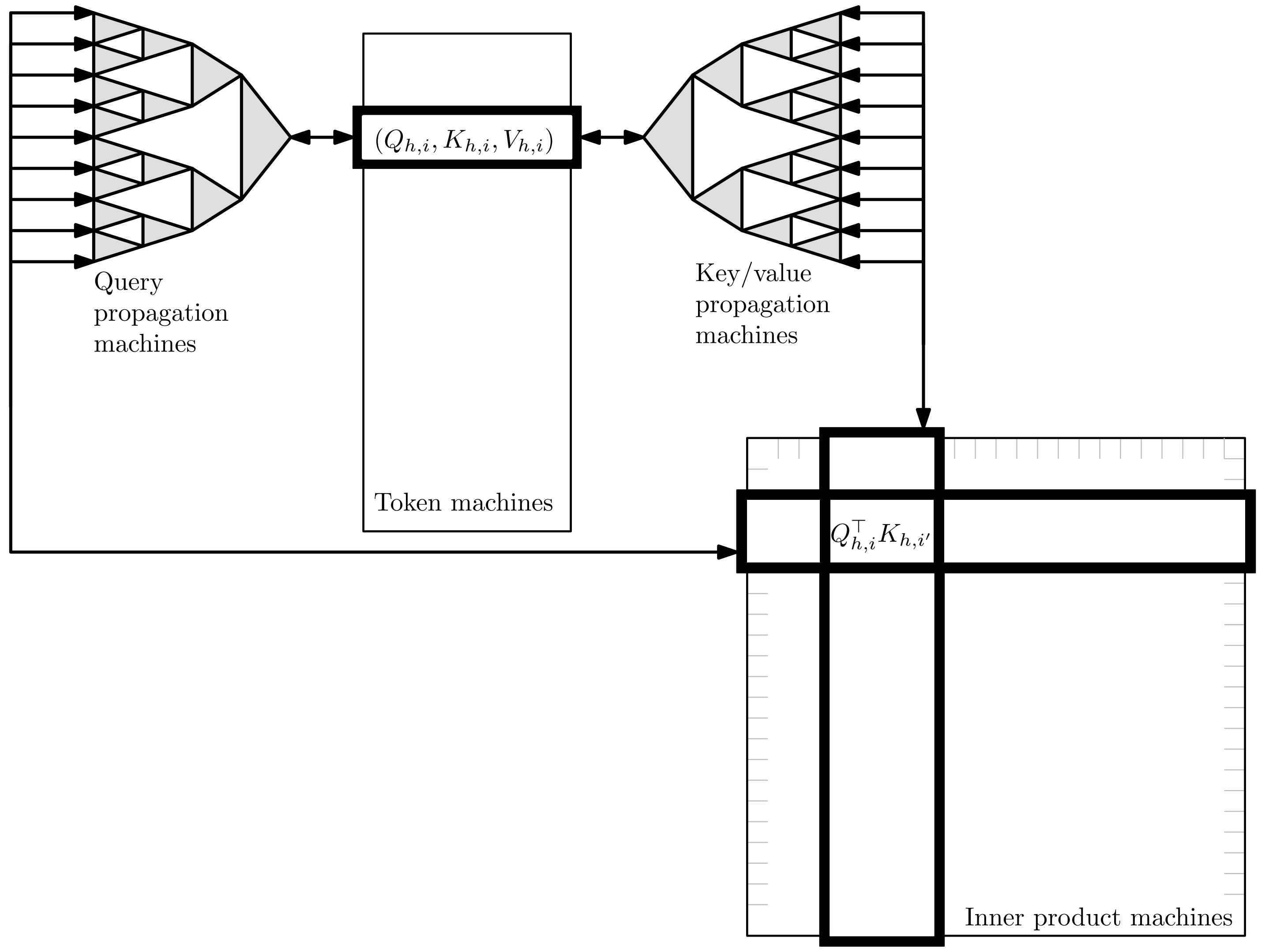


Component 1: Transformer/MPC relationship

Transformers simulate MPC [Improved!]:
R-round MPC protocols with local memory s, # machines q can be simulated by transformers of depth $L = O(R)$, width $m = \tilde{O}(s^{1+\epsilon} \log q)$.

MPC simulates transformers [SHT24]:
Transformers with depth L, width m can be simulated by MPC protocols with $R = O(L)$ rounds, local memory $s = O(m)$, # machines $q = O(N^2)$.

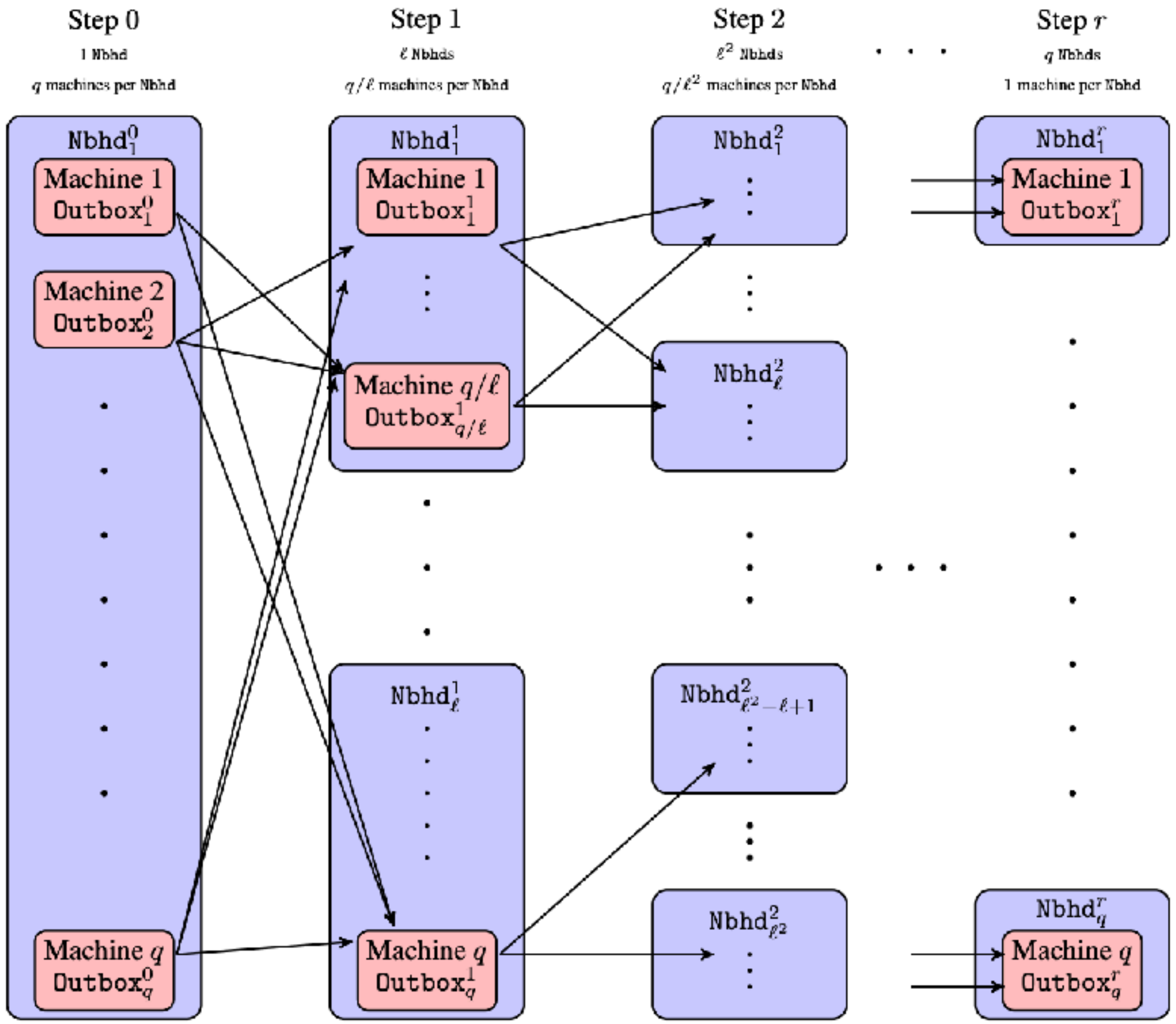




Component 1: Transformer/MPC relationship

Transformers simulate MPC [Improved!]: R-round MPC protocols with local memory s, # machines q can be simulated by transformers of depth $L = O(R)$, width $m = \tilde{O}(s^{1+\epsilon} \log q)$.

MPC simulates transformers [SHT24]: Transformers with depth L, width m can be simulated by MPC protocols with $R = O(L)$ rounds, local memory $s = O(m)$, # machines $q = O(N^2)$.

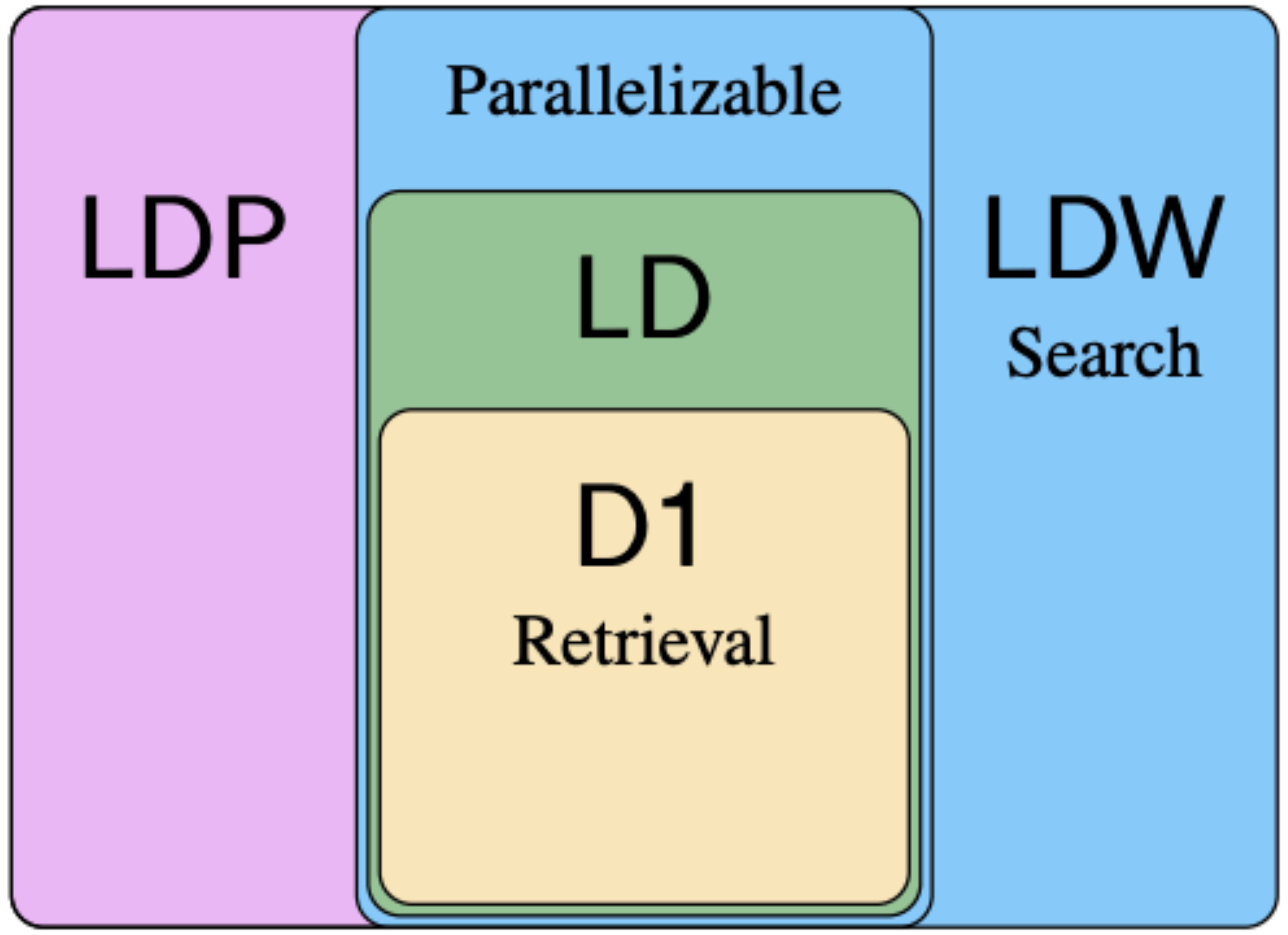


Log-depth proof ideas

Component 1: Bidirectional relationship between transformers and MPC distributed computing model [**S**-Hsu-Telgarsky ’24].

Component 2: Equivalence classes of graph algorithmic tasks in MPC model [Nanongkai-Scquizzato ’22].

Task class	Example tasks	Complexity
Retrieval (§3.3)	Node count	D1
$L = 1$	Edge count	D1
$m = O(\log N)$	Edge existence	D1
	Node degree	D1
Parallelizable (§3.1)	Connectivity	LD
$L = O(\log N)$	Cycle check	LDP∩ LDW
$m = O(N^{\epsilon})$	Bipartiteness	LDP∩ LDW
Search (§3.2)	Shortest path	LDW
$L = O(\log N)$	Diameter	LDW
$m = O(N^{1/2+\epsilon})$		



Log-depth theoretical results

Parallelizable LDW construction: Transformers of depth $L = O(\log N)$ and width $m = O(N^{0.5+\epsilon})$ can solve any parallelizable task.

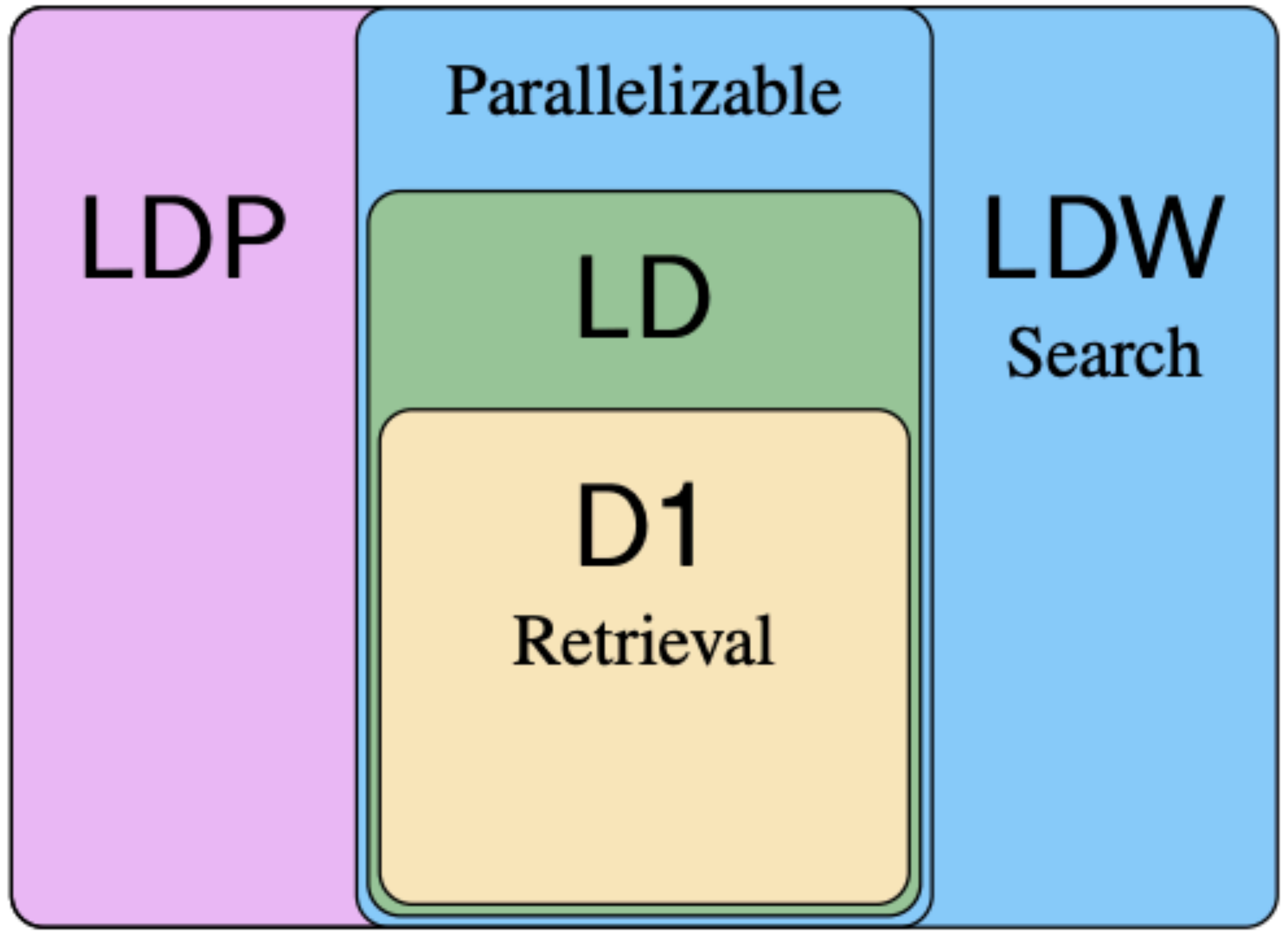
Parallelizable LDP construction: Transformers of depth $L = O(\log N)$ and width $m = O(N^{\epsilon})$ with $N' = N^{O(1)}$ blank input tokens can solve any parallelizable task.

Parallelizable log-depth optimality result: All transformers of width $m = O(N^{1-\epsilon})$ and $N' = N^{O(1)}$ blank tokens that solve any parallelizable task have depth $L = \Omega(\log N)$.

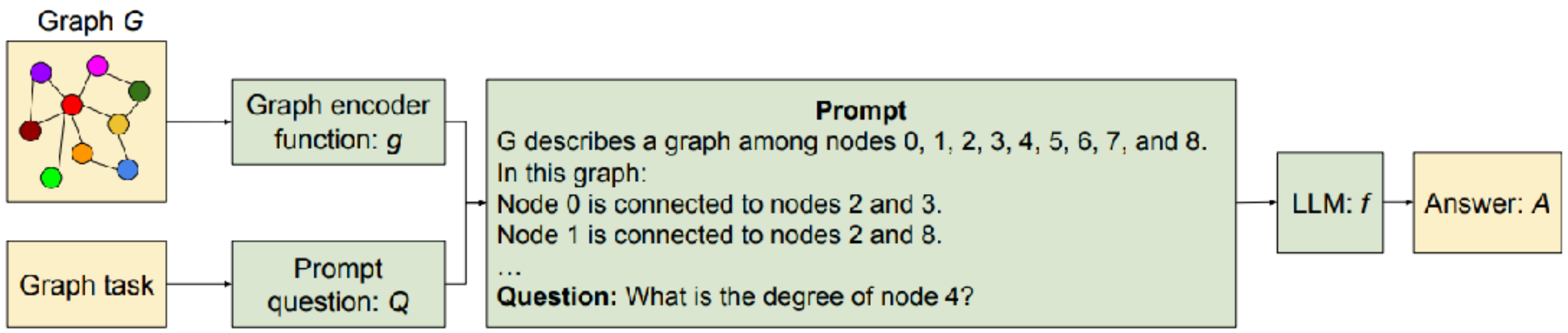
Search LDW construction: Transformers of depth $L = O(\log N)$ and width $m = O(N^{0.5+\epsilon})$ can solve any search task.

Search depth equivalence: If one search task can be solved by transformers of depth L, width $m = N^{O(1)}$, and $N' = N^{O(1)}$ pause tokens, then all search tasks can be solved with depth $L + O(1)$, width $O(m)$ and $O(N') + N^{O(1)}$ phase tokens.

Task class	Example tasks	Complexity
Retrieval (§3.3)	Node count	D1
$L = 1$	Edge count	D1
$m = O(\log N)$	Edge existence	D1
	Node degree	D1
Parallelizable (§3.1)	Connectivity	LD
$L = O(\log N)$	Cycle check	LDP∩ LDW
$m = O(N^{\epsilon})$	Bipartiteness	LDP∩ LDW
Search (§3.2)	Shortest path	LDW
$L = O(\log N)$	Diameter	LDW
$m = O(N^{1/2+\epsilon})$		



Experiments



GraphQA dataset:

- Suite of algorithmic tasks on small graph instances (5-20 nodes).
- Originally designed for "Talk Like a Graph" evaluation of LLM prompting strategies [Fatemi-Halcrow-Perozzi '24].

Models/training regimes:

- 60M-parameter vanilla transformer trained from scratch.
- 11B-parameter pre-trained LLM fine-tuned on GraphQA.
- 62B PaLM LLMs with different prompting strategies [FHP24].
- Various GNN models, including hybrid models used in GraphToken paper [Perozzi, et al '24].
- Sample complexities: 1k vs 100k graph instances.

Experiments

Takeaway #1: GNNs more effectively extract local structure from graphs than transformers in a sample-efficient manner.

- While transformers can efficiently solve these tasks, GNNs have nice inductive biases for “local” solutions.

	Node Degree		Cycle Check	
Model	**1K**	**100K**	**1K**	**100K**
GCN [42]	9.8	9.4	83.2	83.2
MPNN [26]	99.4	**99.8**	99.0	**100.0**
GIN [82]	36.2	37.8	98.8	83.2
60M transformer	31.6	91.7	97.1	98.0
11B transformer (FT)	68.8	—	98.0	—

Experiments

Takeaway #2: While GNNs learn better heuristics with few samples for paralellizable tasks, transformers make better use of more samples.

- Reflects gap in representational capabilities of transformers and GNNs on these tasks.

- GNN inductive biases help when there aren’t enough samples actually solve the task.

Connectivity Task

	# of training samples	
Model	**1K**	**100K**
GCN [42]	50.2	55.0
MPNN [26]	66.8	72.6
GIN [82]	54.0	58.6
60M transformer	57.4	**97.1**
11B transformer (FT)	92.8	—

Experiments

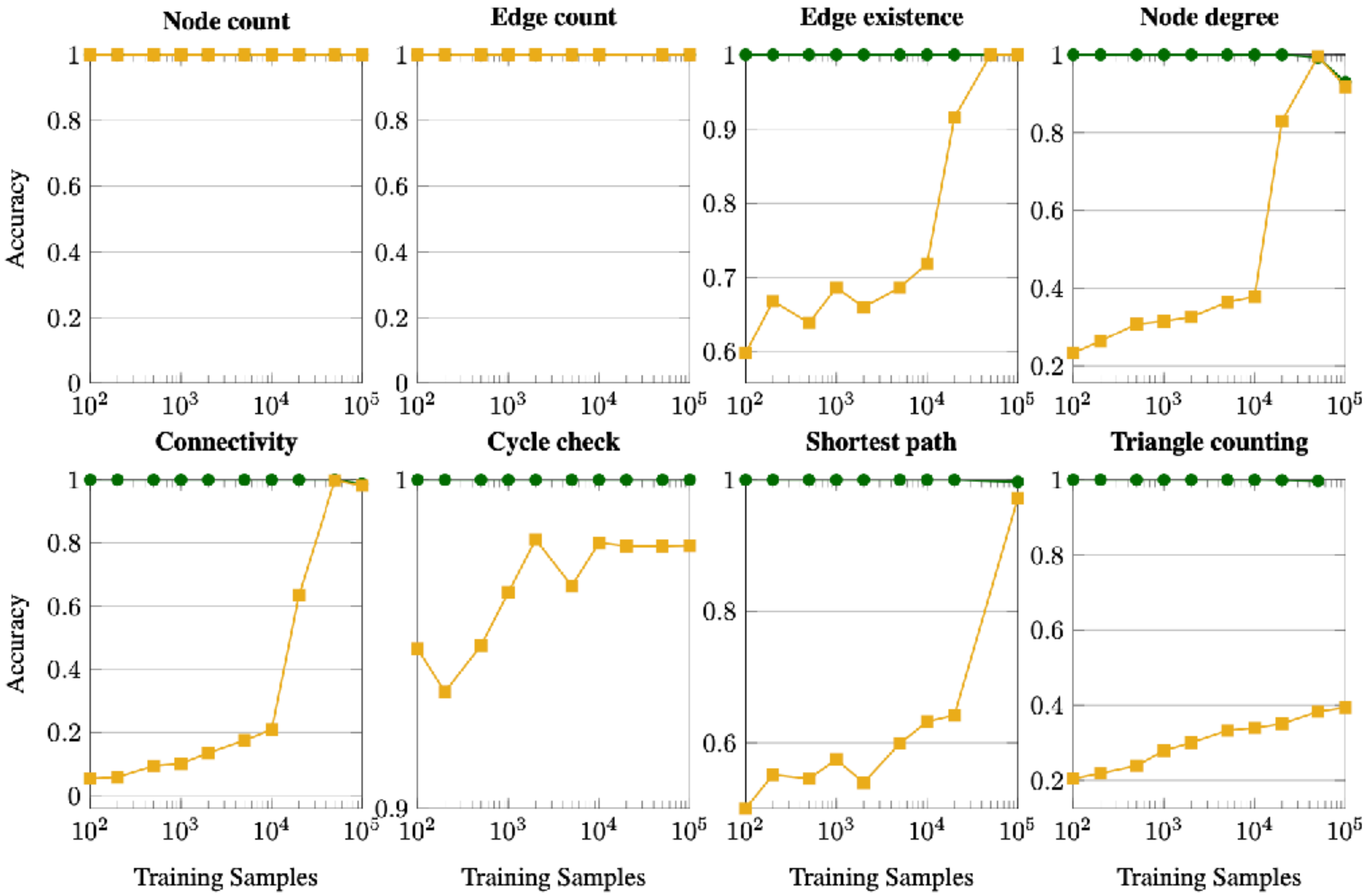
Takeaway #3: Dominance of randomly-initialized and fine-tuned transformers over prompting-based strategies on LLMs.

		Retrieval tasks				Parallelizable Tasks		Search Tasks	Subgraph Counting
	Method	Node count	Edge count	Edge existence	Node degree	Connectivity	Cycle check	Shortest path	Triangle counting
Prompting	ZERO-SHOT [22]	21.7	12.4	44.5	14.0	84.9	76.0	11.5	1.5
	ZERO-COT [22]	14.6	9.4	33.5	10.4	73.5	32.3	33.6	12.7
	FEW-SHOT [22]	25.3	12.0	36.8	17.4	79.4	37.4	22.7	3.0
	COT [22]	27.6	12.8	42.8	29.2	45.2	58.0	38.6	8.1
	COT-BAG [22]	26.9	12.5	37.3	28.0	45.2	52.1	40.4	8.1
Ours	60M transformer-1K	100.0	100.0	67.6	31.5	92.9	97.1	57.4	33.4
	60M transformer-100K	**100.0**	**100.0**	96.1	**91.7**	98.0	**98.0**	97.2	**40.5**
	11B transformer (FT)-1K	**100.0**	45.0	**100.0**	68.8	**98.4**	**98.0**	**92.8**	26.0

- LLMs aren't incidentally learning to solve these tasks on their datasets.

Experiments

Sample complexity experiments for each task with 60M transformers:



What’s next?

- Fine-grained exploration of task difficulty in larger graph instances and size generalization.
- Generalizations of theoretical results and connections to computational complexity.
- Re-examining assumption of arbitrary MLPs.

Thank you