Why does depth help Transformers?

How can a Transformer combine many pieces of information in only a few layers?

Transformers, Parallel Computation, and Logarithmic Depth

Clayton Sanford, Daniel Hsu, Matus Telgarsky

ICML 2024

Turing completeness is not an efficiency bound

Previous presentation	This presentation
Fixed decoder, repeated calls.	Depth scales with input.
Growing output history.	Small token representations.
Exact rational arithmetic.	Parallel token interactions.

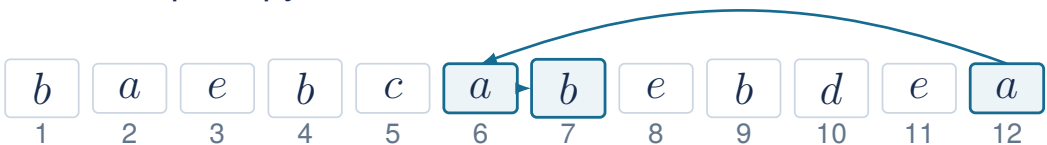
Computability does not imply small depth.

Pérez et al. (2021), Theorem 6 and Section 5; Sanford et al. (2024), Section 1.

Use a copying task to study depth

To compare depths, the paper uses a simple task whose number of repeated lookups can be increased.

One lookup: copy what followed the last earlier match.



Earlier $a\,b$ at 6–7: copy b for the final a.

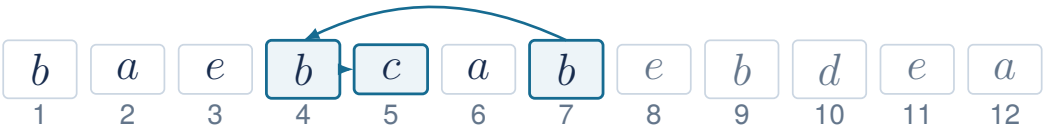
Next: repeat this rule to create dependent lookups.

Sections 1.1 and 4 (pp. 1–2, 5–6). The paper uses this synthetic task for its construction, experiments, and architecture comparisons.

Does each lookup require another layer?

Repeat from b at 7, using only the prefix ending there.

The first lookup determines where the second starts.



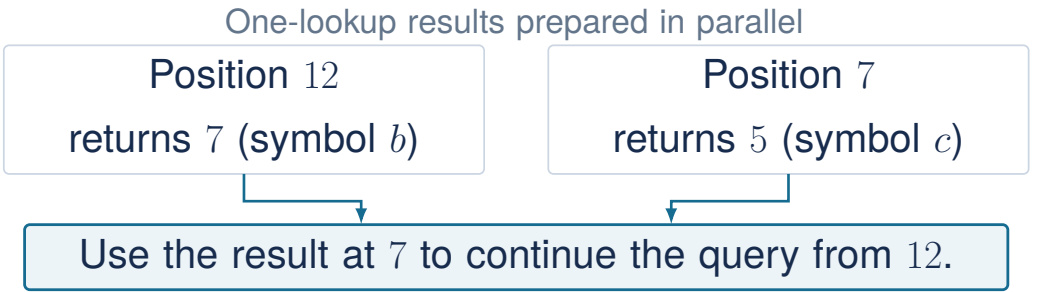
Two lookups: $12 \longmapsto 7 \longmapsto 5$, ending at c.

Let k count the lookups.
How many layers does a small Transformer need?

Definition 4.1 and Section 4.1 (p. 6). The direct execution motivates a question; it is not a depth lower-bound argument.

Can parallel work shorten the chain?

Prepare each position's one-lookup answer in parallel.



To reuse these answers, positions must communicate.

Sections 1.1 and 4.1; Appendix D.1 (pp. 1–2, 6, 26–27). Same sequence; an informal preview, not a layer-by-layer construction.

Prepare one answer at every position

For each position: last earlier match, then the symbol after it.

Position	1	2	3	4	5	6	7	8	9	10	11	12
Input symbol	b	a	e	b	c	a	b	e	b	d	e	a
Last earlier match	–	–	–	1	–	2	4	3	7	–	8	6
Returned position	–	–	–	2	–	3	5	4	8	–	9	7
Returned symbol	–	–	–	a	–	e	c	b	e	–	b	b

– means no earlier match, so there is no one-lookup answer.

All columns use the input, not another column's prepared answer.

Definition 4.1 (p. 6); Appendix D.1 (p. 26). Table computed for the paper's sequence; an informal preview of the prepared results.

How we will answer the depth question

1. **Define what a layer can do.**
   Specify local computation, communication, and storage.
2. **Connect attention to parallel computation.**
   Obtain general algorithms and depth bounds.
3. **Return to the same copying task.**
   Show how prepared answers reduce the number of layers.
4. **Use that task to compare models.**
   Study trained Transformers and limits of other architectures.

Teaching organization following Sections 2–5. The same copying task links the direct construction, experiments, and model separations.

One input row for each position

The input X has N positions, one row X_i per position.

Position 1	Position 2	Position 3
$X_1 = (2)$	$X_2 = (8)$	$X_3 = (5)$
one input number	one input number	one input number

d_{in} is the number of coordinates in each input row.

Inside the model, each position stores m coordinates.

Here $N = 3$ and $d_{\text{in}} = 1$. The working width m may differ.

Section 2.1, Definition 2.2 (p. 3). Three-position input: original teaching example.

The input map ϕ builds a working vector

The input map $\phi_i : \mathbb{R}^{d_{\text{in}}} \to \mathbb{R}^m$ acts on row i.

$$X_i^0 = \phi_i(X_i), \qquad X^0 = \phi(X)$$

Example with $m = 3$: $\phi_i((u)) = (u, i, 0)$.

$(2, 1, 0)$	$(8, 2, 0)$	$(5, 3, 0)$

The coordinates store: input value, position, empty slot.

No row reads another row during initialization.

Section 2.1, Definition 2.2 (p. 3). Per-row dimensions follow $X^0 = \phi(X)$. The coordinate choice is a teaching example.

One attention head: query, key, and value

Write $x_i \in \mathbb{R}^m$ for the current vector at position i.

$$q_i = Q_i(x_i) \qquad k_i = K_i(x_i) \qquad v_i = V_i(x_i)$$

Query: what this position asks for.

Key: how other positions identify this row.

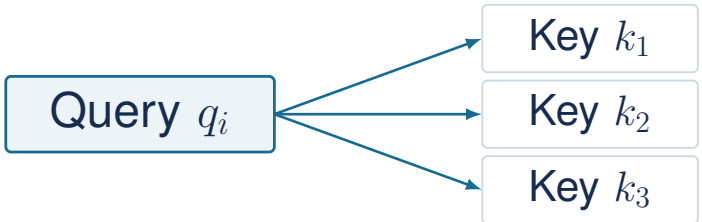
Value: the information this row can send.

$q_i, k_i, v_i \in \mathbb{R}^m$. Each map reads only the current row at i.

Definition 2.1 (p. 3). The maps may be nonlinear and index-specific in this computational model.

A query scores the candidate source rows

Compare query q_i with the key at every source position.



$s_{ij} = \langle q_i, k_j \rangle$ is one scalar score.

i: receiving position. j: candidate source position.

Section 2.1, Definition 2.1 (p. 3).

Softmax turns scores into weights

For the fixed query i, normalize its N scores with softmax.

$$a_{ij} = \frac{\exp(s_{ij})}{\Sigma_{r=1}^{N} \exp(s_{ir})}$$

a_{ij} is the weight assigned to the value at source position j.

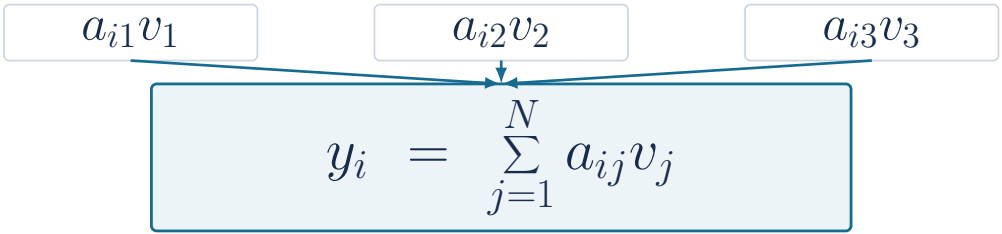
$$a_{ij} \geq 0, \qquad \Sigma_{j=1}^{N} a_{ij} = 1.$$

The index r in the denominator runs over all source positions.

Section 2.1, Definition 2.1 (p. 3).

An attention head returns one vector per row

The values $v_j \in \mathbb{R}^m$ carry the information to combine.



The result y_i is one m-dimensional vector, not N vectors.

Every receiving position computes its own weighted combination.

Section 2.1, Definition 2.1 (p. 3).

Equal scores give an ordinary average

Two sources, with values $v_1 = 2$ and $v_2 = 8$.

Scores	$s_{i1} = 0, \qquad s_{i2} = 0$
Weights	$a_{i1} = a_{i2} = \frac{e^0}{e^0 + e^0} = \frac{1}{2}$
Output	$y_i = \frac{1}{2} \cdot 2 + \frac{1}{2} \cdot 8 = 5$

Equal scores give an average of the two values.

Section 2.1, Definition 2.1 (p. 3). Original two-source teaching example; all scores are equal, with no omitted sources.

Why the averaging example matters

Can we recover both messages from their average?

$$\frac{2+8}{2} = 5 \qquad \frac{4+6}{2} = 5$$

Different message pairs produce the same output.

An average is not a message list.

The routing proof will encode the messages before combining them.

Lemma 3.2 and Appendix B.1 (pp. 5, 15–17). Scalar pairs are original teaching examples.

The function f names the whole attention head

Let $Z \in \mathbb{R}^{N \times m}$ collect the current working vectors.
$Q(Z)$, $K(Z)$, and $V(Z)$ collect their queries, keys, and values.

$$f_{Q,K,V}(Z) \;=\; \mathrm{softmax}\Big(Q(Z)K(Z)^{\top}\Big)V(Z)$$

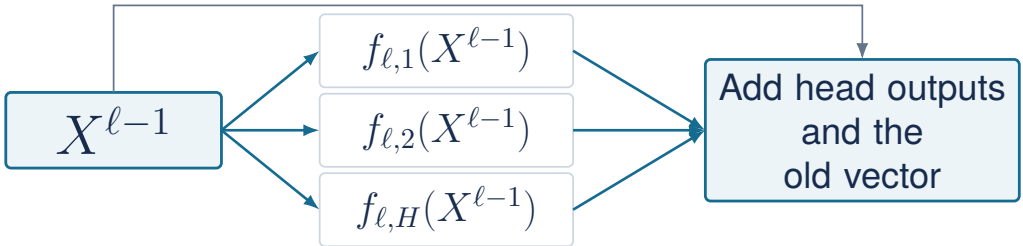
$$f_{Q,K,V} \;:\; \mathbb{R}^{N \times m} \;\longrightarrow\; \mathbb{R}^{N \times m}$$

f includes scoring, softmax,
and the weighted combination.

Section 2.1, Definition 2.1 (p. 3).

Several heads still make one attention layer

Layer ℓ has H heads. Head h is written $f_{\ell,h}$.
All read $X^{\ell-1}$: the working vectors before layer ℓ.



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

The $+X^{\ell-1}$ term is the residual connection.

Section 2.1, Definition 2.2 (p. 3).

The output MLP ψ reads the final vector

The output MLP acts on one final vector X_i^L at a time.

$$\psi_i : \mathbb{R}^m \longrightarrow \mathbb{R}^{d_{\text{out}}}$$
$$T(X)_i = \psi_i(X_i^L)$$

Example: read the third coordinate, $\psi_i(a, b, c) = c$.

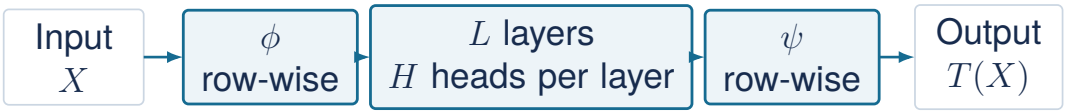
$$(2, 1, 5) \longmapsto 5$$

MLP = multi-layer perceptron. d_{out} counts output coordinates.

Section 2.1, Definition 2.2 (p. 3). Output-coordinate projection is an original teaching example.

The complete Transformer architecture

Embed the input, apply L attention layers, read the output.



$$X^0 = \phi(X)$$

$$X^\ell = X^{\ell-1} + \Sigma_{h=1}^{H} f_{\ell,h}(X^{\ell-1}), \quad 1 \leq \ell \leq L$$

$$T(X) = \psi(X^L)$$

Section 2.1, Definition 2.2 (p. 3).

Causal attention cannot read the future

Query i reads only positions $j \le i$.

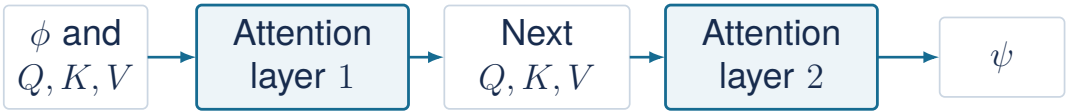
	$j=1$	2	3	4
$i=1$	✓			
2	✓	✓		
3	✓	✓	✓	
4	✓	✓	✓	✓

Allowed rows are still updated in parallel.

Section 2.1, masked Transformers (p. 3). A check marks an allowed attention entry.

Depth counts attention stages in this model

L counts attention layers, not local arithmetic.



The row-wise maps may be arbitrary functions in this paper.

Their internal depth and running time are not included in L.

Definition 2.2 and modeling assumptions (p. 3); Appendix B.2, Lemma B.5 (p. 20).

The stored information is still bounded

Each row has m coordinates. Each coordinate uses p bits.



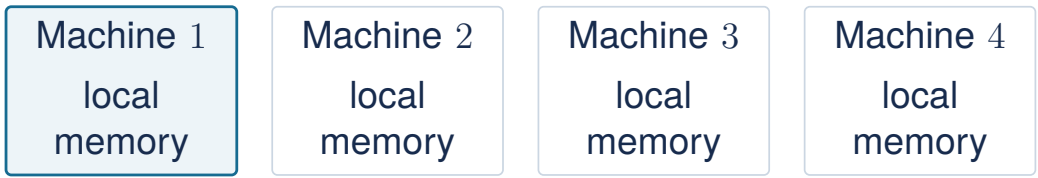
$$p = \Theta(\log N), \qquad O(mp) \text{ bits stored per row}$$

A local map receives only the information stored in its row.

Section 2.1, modeling assumptions; Appendix A.1 (pp. 3, 13–14).

Massively Parallel Computation

Split the input across q machines.



Each machine stores at most s words.

A word has $p = \Theta(\log n)$ bits; n is the input word count.

The picture shows four machines, not a restriction on q.

Definition 2.3 (p. 3). MPC = Massively Parallel Computation.

A round has a local phase and a routing phase

1. Compute any function of the current local memory.
2. Prepare addressed messages totaling at most s words.
3. Transmit all machines' messages simultaneously.
4. Every receiver gets at most s words.

Definition 2.3 and Figure 1 (pp. 3–4).

Local memory belongs to one machine

Take $q = 8$ machines, with $s = 4$ word slots in each.



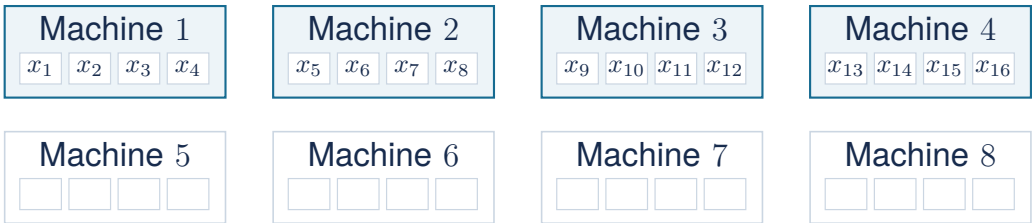
Local capacity: 4 words at one machine.

Total capacity: $8 \times 4 = 32$ words across the system.

Definition 2.3 (p. 3). Eight-machine, four-word capacities are a teaching example; each small cell denotes one word.

Extra total memory is additional workspace

Place the 16-word input in the same eight machines.



32 slots = 16 input words + 16 initially unused slots.
Each machine still holds only 4 words.

Definition 2.3 (p. 3): the input starts in the first $\lceil n/s \rceil$ machines. The numerical layout is a teaching example.

Two exponents set the memory budget

Let n be the input word count. Fix constants $\delta, \gamma > 0$.

δ controls one machine: $s = O(n^{\delta})$ words

Number of machines: $q = \Theta(n^{1+\gamma-\delta})$

γ controls the total: $qs = O(n^{1+\gamma})$ words

Input size: n words. Total-space budget: $O(n \cdot n^{\gamma})$ words.

Example: $n = 16,\ \delta = \frac{1}{2},\ \gamma = \frac{1}{4}$ gives $s = 4,\ q = 8$.

Definition 2.3 (p. 3). $qs = O(n^{1+\gamma})$ follows by multiplying its bounds. Numerical example chooses unit constants.

Memory bounds use words, not bits

One word contains $p = \Theta(\log n)$ bits.

Quantity	Words	Bits
Input	n	$np = \Theta(n \log n)$
One machine	$s = O(n^{\delta})$	$sp = O(n^{\delta} \log n)$
All machines	$qs = O(n^{1+\gamma})$	$qsp = O(n^{1+\gamma} \log n)$

q counts machines. It is not an amount of memory.

Definition 2.3 and its word-size footnote (p. 3). Bit bounds are the stated word bounds multiplied by p.

Two directions, two uses



Transfer parallel algorithms into attention layers.



Transfer conjectured MPC hardness into depth lower bounds.

Theorems 3.1 and 3.4 (pp. 4–5).

A working vector represents one machine

Assign machine i to sequence position i.



Example: a four-word memory $(2, 2, 9, 0)$.

$$X_i^\ell \;=\; (2, 2, 9, 0, 0, \ldots, 0) \;\in\; \mathbb{R}^m$$

The proof also reserves coordinates for routing.

A token here means a vector, not one scalar.

Appendix B.2 (p. 18), machine-to-position assignment. The displayed padded vector is a schematic teaching layout.

Compute outgoing messages locally

The maps Q, K, V apply these example rules locally.

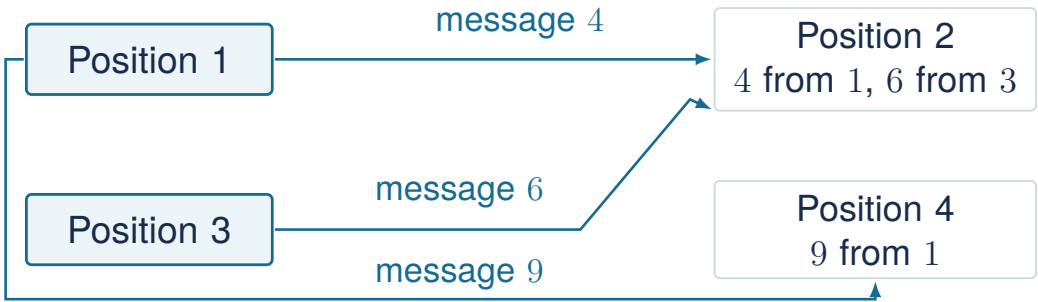
Machine 1	Machine 3
Memory: $(2, 2, 9, 0)$	Memory: $(1, 5, 0, 0)$
Send $4 = 2 + 2$ to 2.	Send $6 = 1 + 5$ to 2.
Send 9 to 4.	
Outgoing list: $(4, 2),\ (9, 4)$	Outgoing list: $(6, 2)$

Each pair is $(\text{message}, \text{destination})$.

Theorem 3.1 overview and Lemma B.5 (pp. 4, 20). Arithmetic rules are a teaching example producing the existing routing packets.

Route messages to their destinations

Route these messages simultaneously.



Required routing behavior; construction follows.

Lemma 3.2 and Appendix B.2 (pp. 5, 19–20). Arrows specify required transfers, not literal attention weights.

From MPC rounds to attention layers

Start with a deterministic R-round MPC protocol.

N input words; at most N output words; $0 < \gamma < \delta < 1$.

$$L = R + 1$$

attention layers reproduce the output on every input.

$$m = O(N^{4\delta} \log N)$$

working coordinates per position

$$H = O(\log \log N)$$

heads per layer

Theorem 3.1 (p. 4). Deterministic protocol; $n_{\text{out}} \leq n_{\text{in}} = N$.

What must a receiver recover?

Receiver 2 must separate the messages intended for it.

Source	Outgoing (message, destination)
1	$(4, 2)$ and $(9, 4)$
3	$(6, 2)$

Required result: $(4, 1)$ and $(6, 3)$.

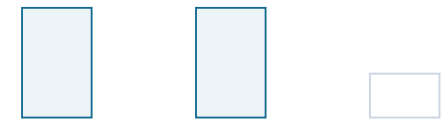
Pairs are (message, source); message 9 is excluded.

Recall: the raw average loses the individual messages.

Lemma 3.2 and Appendix B.1 (pp. 5, 14–17). Numerical messages are a teaching example.

From softmax weights to exact primitives

Give the desired rows the same top score.



Other rows have
a smaller score.

Scale scores to suppress the unwanted softmax weight.

Lemma A.2 gives the desired output after rounding.
Exactness uses the paper's specified finite-precision model.

Appendix A.1, Lemma A.2, and Appendix F.1 (pp. 13, 32–33). Bars schematically show tied maxima and a separation margin.

First select the sources that contact me

Source j stores an encoding z_j and a destination set S_j.

$$Y_i \quad = \quad \frac{1}{b_i} \Sigma_{j:\, i \in S_j} \, z_j$$

b_i: number of sources sending to receiver i.

With no sender, a dummy token returns zero.

All receivers select their sources in one head.

Proposition B.1 and Appendix F.2 (pp. 14, 33–34).

What must survive the attention average?

1. Encode locally → 2. Average once → 3. Decode locally

Zoom in on receiver 2: it needs two separate messages.

Source 1

$$4 \longrightarrow 2$$
$$9 \longrightarrow 4$$

Source 3

$$6 \longrightarrow 2$$

Each arrow means: message $\longrightarrow$ destination.

Receiver 2: 4 from 1, and 6 from 3

Encode locally, use one attention average, then decode locally.

Lemma 3.2; Appendix B.1, Figure 3 (pp. 14–17). Same numerical routing example as the preceding slides.

Read the four fields of one packet

1. Encode locally → 2. Average once → 3. Decode locally

Turn one addressed message into a four-field **packet**.

Message	Source	Destination	Validity α
4	1	2	1

$(4, 1, 2, 1)$ means: message 4, from 1, for 2, present.

0	0	0	0

An empty slot stores $(0, 0, 0, 0)$.

Before averaging: $\alpha = 1$ for a packet; $\alpha = 0$ for empty.

Appendix B.1, packet representation (p. 15). The payload is one number in this teaching example.

A slot is part of a vector, not a machine

1. Encode locally → 2. Average once → 3. Decode locally

A **slot** is a block of four coordinates in a value vector.

z_1: one vector prepared at source 1

slot a	slot b	slot c	slot d	slot e	slot f
1–4	5–8	9–12	13–16	17–20	21–24

Coordinate numbers

$$z_{1,a} = (4, 1, 2, 1) \text{ fills coordinates } 1\text{–}4.$$

Slot a means coordinates 1–4 at every source.
Slots are not machines, heads, or time steps.

Appendix B.1 (pp. 15–16). Six four-coordinate blocks display the packet data; selection codes are separate.

Place copies in a shared slot layout

1. Encode locally → 2. Average once → 3. Decode locally

Tuple order: (message, source, destination, validity).

Slot	Source 1: vector z_1	Source 3: vector z_3
a	$(4,1,2,1)$	$(0,0,0,0)$
b	$(0,0,0,0)$	$(6,3,2,1)$
c	$(4,1,2,1)$	$(6,3,2,1)$
d	$(9,1,4,1)$	$(0,0,0,0)$
e	$(0,0,0,0)$	$(0,0,0,0)$
f	$(4,1,2,1)$	$(0,0,0,0)$

Message 4 uses a, c, f. Copy locations are fixed in advance.

Appendix B.1 and Fact B.2 (pp. 15–17). Teaching layout: original slots a–d, plus empty e and duplicate f.

Average the two source vectors

1. Encode locally → **2. Average once** → 3. Decode locally

Receiver 2 selects sources 1 and 3, because both send to 2.

Source 1: $S_1 = \{2, 4\}$ Source 3: $S_3 = \{2\}$

Whole value vector z_1 Whole value vector z_3

$$Y_2 = \frac{1}{2} z_1 + \frac{1}{2} z_3$$

Two selected source vectors: divide by 2, not by 3 messages.

Source 1's packet for receiver 4 is included too.

Proposition B.1 and Appendix B.1 (pp. 14–16). Averaging primitive uses Lemma A.2 as explained earlier.

A clean slot preserves one scaled packet

1. Encode locally → 2. Average once → 3. Decode locally

Zoom into slot a: only source 1 placed a packet here.

Same slot	Message	Source	Destination	Validity
Source 1	4	1	2	1
Source 3	0	0	0	0
Receiver 2	$\frac{4+0}{2} = 2$	$\frac{1+0}{2} = \frac{1}{2}$	$\frac{2+0}{2} = 1$	$\frac{1+0}{2} = \frac{1}{2}$

One packet plus zeros gives a scaled copy of that packet.
Averaged source $1/2$ is not an address; rescale it first.

Appendix B.1 (p. 16), clean slots. Each of the four coordinate calculations is shown explicitly.

A collision mixes two different packets

1. Encode locally → 2. Average once → 3. Decode locally

Zoom into slot c: both sources placed a packet here.

Same slot	Message	Source	Destination	Validity
Source 1	4	1	2	1
Source 3	6	3	2	1
Receiver 2	$\frac{4+6}{2} = 5$	$\frac{1+3}{2} = 2$	$\frac{2+2}{2} = 2$	$\frac{1+1}{2} = 1$

Validity 1 now means two packets collided in this slot.
Clean copies in other slots still preserve 4 and 6.

Appendix B.1 (p. 16), corrupted slots. The tuple (5,2,2,1) is an average, not a newly sent packet.

Inspect the one vector that receiver 2 gets

1. Encode locally → 2. Average once → **3. Decode locally**

The decoder receives Y_2 only; it cannot inspect z_1 or z_3.

Slot	Message coordinate	Source coordinate	Destination coordinate	Validity $\bar{\alpha}$	Meaning
a	2	$1/2$	1	$1/2$	one packet
b	3	$3/2$	1	$1/2$	one packet
c	5	2	2	1	collision
d	$9/2$	$1/2$	2	$1/2$	one packet
e	0	0	0	0	empty
f	2	$1/2$	1	$1/2$	one packet

Appendix B.1 (pp. 16–17). All entries are computed from the six-slot teaching layout, including empty e and duplicate f.

ψ is the output MLP for decoding

MLP = multi-layer perceptron; ψ_i acts at receiver i.



The construction prescribes a fixed decoding program.

It does not train this decoder on example messages.

It reads all slots of Y_2, but no other token's vector.

Definition 2.2 and modeling assumptions (p. 3); Appendix B.1, proof of Lemma 3.2 (pp. 14, 16–17).

The decoding program assigned to ψ_i

At receiver i, the MLP is prescribed to compute this program:

```
if every marker is zero: return an encoded empty set
scale = smallest positive validity marker
for each slot whose marker equals scale:
   recover message, source, destination by dividing by scale
   if destination equals i: retain (message, source) once
return a vector encoding of the retained pairs
```

The rule is fixed. The scale and retained pairs depend on Y_i.

Appendix B.1, decoder (p. 17); routing set output (pp. 4–5). Pseudocode states the required function, not neural weights.

The MLP capacity is an assumption here

The authors claim that an MLP can compute this decoder.

Their explicit model treats MLPs as:

> “arbitrary functions
> on the embedding space”

Its hidden width and depth are not bounded in this model.

Hidden layers and numerical weights are not specified.

m bounds the token vector, not hidden neurons inside ψ.

Definition 2.2, modeling assumptions (p. 3), and footnote 2 (p. 4); Appendix B.1, decoder existence and rule (pp. 16–17).

Recover the clean packets first

Here the smallest positive validity marker is $1/2$.

$1/2$: clean

1: collision

0: empty

Slot a	Message	Source	Destination	Validity
Received	2	$1/2$	1	$1/2$
$\times 2$	↓	↓	↓	↓
Restored	4	1	2	1

Restore a, b, d, f this way. Ignore the collision c and empty e.

Appendix B.1, facts 1–3 and decoder (p. 17). Marker values shown are specific to this two-source example; the clean-copy guarantee is used.

Keep messages for receiver 2, once each

Now read the restored packets as ordinary messages.

Slot	Recovered packet	Decision at receiver 2
a	4 from 1, to 2	Keep
b	6 from 3, to 2	Keep
d	9 from 1, to 4	Not for 2
f	4 from 1, to 2	Copy of a

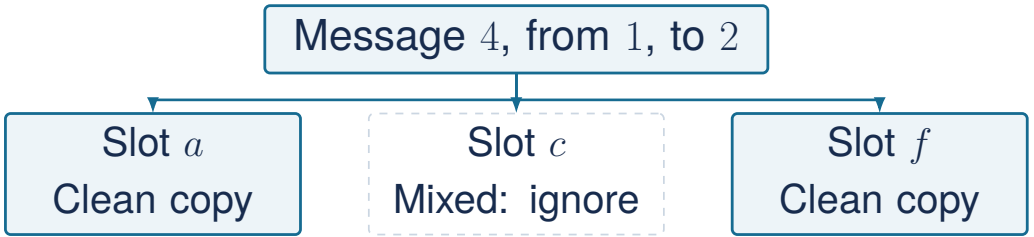
Output: 4 from 1 and 6 from 3.

Same message and same source means one output, not two.

Routing output is a set of (message, source) pairs (pp. 4–5); Appendix B.1, destination check (p. 17). Same teaching packets.

Why can we ignore the mixed slots?

Copy each packet into carefully chosen slots.



Guarantee from the paper

Every message has a clean copy.

ψ reads clean copies; it does not reconstruct mixed packets.

Fact B.2 and Appendix B.1 (pp. 16–17). This picture shows one packet from the example; the fixed-slot proof is in the technical backup.

First: is this slot usable?

1. Check the marker | 2. Check the address | 3. Remove copies

The validity marker tells us whether a slot is usable.

Here we average two sources: each contributes a marker 1 or 0.

Slot a

$$\frac{1+0}{2} = \frac{1}{2}$$

Clean: restore

Slot c

$$\frac{1+1}{2} = 1$$

Mixed: ignore

Slot e

$$\frac{0+0}{2} = 0$$

Empty: ignore

The smallest positive marker is $1/2$: keep those slots.

So restore a, b, d, f; do not try to repair c or read e.

Appendix B.1, clean/corrupted/blank slots and decoder (pp. 16–17). Marker values are for the two-source teaching example.

Next: is this packet for me?

1. Check the marker | **2. Check the address** | 3. Remove copies

Receiver 2 checks the destination of each restored packet.

Restored slot a

Message	From	To
4	1	2

$2 = 2$ Keep

Restored slot d

Message	From	To
9	1	4

$4 \neq 2$ Discard

A clean packet can still be addressed to another receiver.
Restore first; compare the address second.

Appendix B.1, destination test in the decoder (p. 17). The displayed destination fields have already been restored.

Finally: have we kept this pair already?

1. Check the marker | 2. Check the address | **3. Remove copies**

Compare both fields: the message and its source.

Slot a	Slot b	Slot f
Message 4 From 1	Message 6 From 3	Message 4 From 1
Keep	Keep	Copy of a: skip

ψ_2 returns: 4 from 1 and 6 from 3.

Equal messages from different sources are not duplicates.

Routing output is a set of (message, source) pairs (pp. 4–5); Appendix B.1 (p. 17). First-copy language is a teaching description.

Attention communicates; ψ decodes locally

Only attention combines different positions.



The next local rule can use both payloads: $4 + 6 = 10$.

Decoding does not require another attention layer.

Between rounds, decoding is folded into the next local maps.

Definition 2.2 and arbitrary row-wise maps (p. 3); Appendix B.2 (pp. 18–20). Residual bookkeeping omitted; addition is illustrative.

Finish the simulation: count the layers

First gather input words into machine representations.



$$1 + (R-1) + 1 = R + 1 \text{ attention layers}$$

Message-size grouping and width details are in the appendix.

Theorem 3.1; Lemmas B.4–B.7 (pp. 4, 18–20).

Connectivity needs only logarithmic depth

D: the largest diameter of a connected component.

Diameter is the largest shortest-path distance within that component.

$$L = O(\log D), \qquad m = O(N^{\epsilon}), \qquad H = O(\log \log N)$$

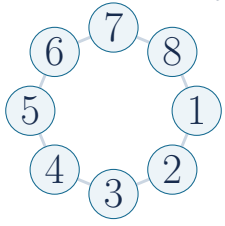
Identify all components, for every fixed $\epsilon \in (0, 1)$.

This upper bound does not assume a conjecture.

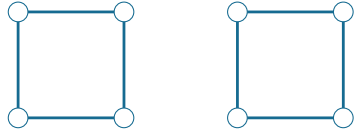
Corollary 3.3 (p. 5), using the MPC algorithm of Coy and Czumaj (2022).

One cycle or two cycles?

Can an MPC protocol distinguish these two kinds of graph?
Sublinear local memory; polynomial total memory.



One N-cycle



Two $(N/2)$-cycles

Conjecture: $\Omega(\log N)$ rounds are needed.

Conjecture 2.4 (p. 4). The drawings are small illustrations; input labels and edge order are arbitrary.

From attention layers back to MPC rounds

Suppose $mH = O(N^{\delta})$. Choose $\delta < \delta\prime < 1$.

$$O(L/(\delta\prime - \delta)) \text{ MPC rounds}$$

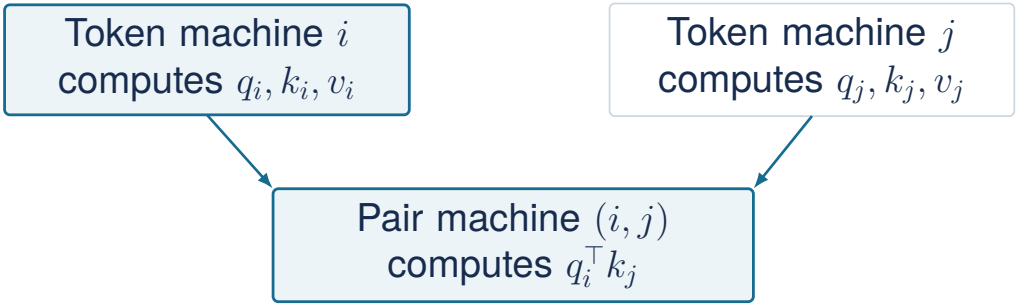
For a fixed exponent gap: $O(L)$ rounds.

$s = O(N^{\delta\prime})$ words	$q = O(N^2)$ machines

Applies to both masked and unmasked Transformers.

Theorem 3.4 (p. 5); Appendix C.1. Applies to masked and unmasked Transformers.

Separate local maps from pairwise scores

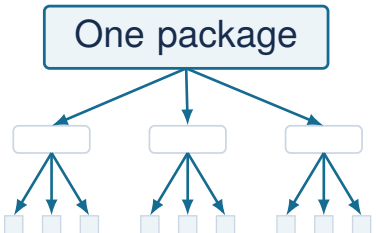


One pair machine for each of the N^2 query–key pairs.

Theorem 3.4 proof overview and Figure 5 (pp. 5, 21).

Bound the copies sent by each machine

Distribute one vector through a tree of small machines.



A package has $O(mH)$ words.

$$b \asymp s/(mH) \text{ copies}$$

Choose b polynomial in N using the exponent gap.

$$O(\log_b N) \text{ tree levels}$$

Only a bounded number of copies leaves each node.

The drawing uses 3 children; the proof chooses b from memory.

Appendix C.1, propagation machines and Figure 5 (pp. 21–23). The drawing illustrates branching, not the asymptotic value of b.

Softmax has aggregatable partial results

For a fixed query i, partition the key positions into groups B.

$$Z_B = \Sigma_{j \in B} e^{q_i^\top k_j}, \qquad U_B = \Sigma_{j \in B} e^{q_i^\top k_j} v_j$$

$$y_i = \frac{\Sigma_B U_B}{\Sigma_B Z_B}$$

The tree combines partial normalizers and weighted outputs.

The finite-precision implementation is analyzed in Appendix C.1.

Appendix C.1, softmax computation and precision analysis (pp. 23–25). Displayed equations are exact-real identities.

Why connectivity needs logarithmic depth

ASSUMING THE ONE-VERSUS-TWO CYCLE CONJECTURE

Fix $\epsilon \in (0,1)$. Component diameters are at most D.

$$D \geq N^{\epsilon}, \qquad mH = O(D^{1-\epsilon})$$

$$L = \Omega(\log D)$$

A shallower solver would contradict the cycle conjecture.

A worst-case bound in the stated representation-size regime.

Corollary 3.5 (p. 5); Appendix C.2. Exact restrictions are shown below.

A direct construction for the sequence task

The MPC simulation handles general parallel algorithms.

The sequence task has more structure.

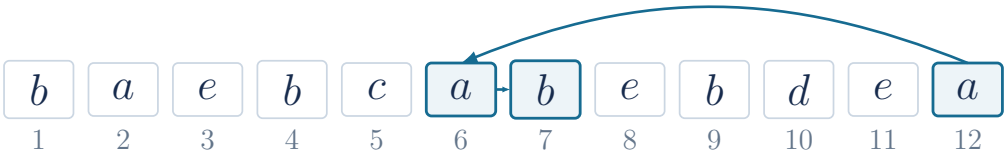
General simulation	Direct construction
Polynomial width	Constant width One head

Not an application of the generic simulation theorem.

Section 4.1, immediately after Theorem 4.2 (p. 6). Contrast with Theorem 3.1.

Copy the token after the last earlier match

The final token is a. Its last earlier occurrence is at position 6.



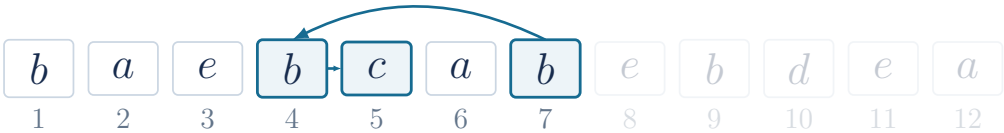
$$12 \longmapsto 7, \qquad \mathrm{hop}_1(X)_{12} = b$$

Return the following token, not the matched token itself.

Section 4 and Definition 4.1 (p. 6). The input sequence is the paper's example.

Repeat the operation at the returned position

Now the current position is 7, with symbol b.



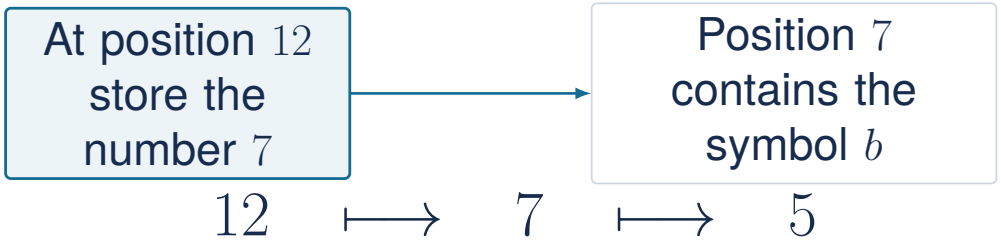
$$12 \longmapsto 7 \longmapsto 5, \qquad \mathrm{hop}_2(X)_{12} = c$$

Do not search the full original sequence again.

Section 4 and Definition 4.1 (p. 6). Same input; the second search uses the prefix ending at position 7.

A pointer is a stored position number

A pointer is a number that identifies a position to read next.



Follow it: read the row at the stored position.

Positions: 7 and 5. Returned symbols: b and c.

Definition 4.1 (p. 6). Terminology illustrated using the two preceding steps on the paper's example.

Separate the position from the symbol

For this task, f denotes the one-hop position map of input X:

$$f(i) \;=\; \max\bigl(\{0\} \cup \{j \;:\; 2 \le j \le i,\; X_{j-1} = X_i\}\bigr).$$

$f^k(i)$ means apply f repeatedly k times.

$$\mathrm{hop}_k(X)_i \;=\; \begin{cases} X_{f^k(i)}, & f^k(i) > 0, \\ \bot, & f^k(i) = 0. \end{cases}$$

Here f differs from $f_{\ell,h}$; $f(0) = 0$ and $f^0(i) = i$.

Definition 4.1 (p. 6). We write f for find^1_X and use an explicit null position 0.

A missing match and a self-pointer are different

Missing match

Here $X_5 = c$.

There is no earlier c.

Hence $f(5) = 0$.

Adjacent repeat

Input: aa.

The earlier a is at 1.

Its following position is 2.

So $f(2) = 2$.

Always $0 \leq f(i) \leq i$. Self-pointers are allowed.

Definition 4.1 and Appendix A.1. This slide follows the inclusive $j \leq i$ convention in the definition.

Following one pointer at a time looks serial

Start instead at position 11 in the same input.

$$11 \xrightarrow{\ e\ } 9 \xrightarrow{\ b\ } 8 \xrightarrow{\ e\ } 4 \xrightarrow{\ b\ } 2 \xrightarrow{\ a\ }$$

Four pointer steps return $\mathrm{hop}_4(X)_{11} = a$.

Prepare longer jumps at all positions simultaneously.

Teaching trace computed from Definition 4.1 on the same paper example.

Logarithmic depth, constant width, one head

For any $k \geq 1$ and alphabet Σ with $|\Sigma| \leq N$:

$$L \;=\; \lfloor \log_2 k \rfloor + 2$$

Causal attention computes every k-hop answer.

$$m \;=\; O(1)$$

coordinates per position

$$H \;=\; 1$$

attention head

$p = \Theta(\log N)$ bits per coordinate; sketch for power-of-two k.

Theorem 4.2 (p. 6). Exact statement is quoted; the worked proof below treats k that are powers of two.

Layer 1: prepare the same rows

b	a	e	b	c	a	b	e	b	d	e	a
1	2	3	4	5	6	7	8	9	10	11	12

Layer 1: every position copies its preceding symbol.

Position i	Current X_i	Previous X_{i-1}
2	a	b
4	b	e
8	e	b
9	b	e
11	e	d

The selected rows stay fixed; all positions update in parallel.

Appendix D.1, first layer; Lemma D.1 (p. 26).

Layer 2: prepare one-hop answers everywhere

b	a	e	b	c	a	b	e	b	d	e	a
1	2	3	4	5	6	7	8	9	10	11	12

Choose the latest $j \le i$ whose preceding symbol is X_i.

Position i	One-hop target $f(i)$	Symbol $X_{f(i)}$
2	0	$\perp$
4	2	a
8	4	b
9	8	e
11	9	b

Each row stores its target and symbol; 0 denotes no match.

Appendix D.1, second layer (p. 26). Selected rows from the running example; inclusive matching follows Definition 4.1.

One lookup returns the target and its symbol

Use the stored target to choose the row to read.



Value: $\big(f(j), X_{f(j)}\big)$

One head returns both the next target and its symbol.

Constructed scores and the rounding lemma justify the lookup.

Lemmas D.1–D.2 and Appendix D.1 (p. 26). These are specially constructed attention maps.

Layer 3: double every stored jump

b	a	e	b	c	a	b	e	b	d	e	a
1	2	3	4	5	6	7	8	9	10	11	12

Layer 3: read a one-hop pointer at your one-hop target.

Row i	Stored $f(i)$	Read row j	Copy $f(j)$	New $f^2(i)$
2	0	0 (null)	0	0
4	2	2	0	0
8	4	4	2	2
9	8	8	4	4
11	9	9	8	8

$f^2(11) = f(9) = 8$. Simultaneously, $f^2(8) = f(4) = 2$.

Appendix D.1, doubling construction (pp. 26–27). Teaching trace for $k = 4$.

Layer 4: reuse the two-hop answers

b	a	e	b	c	a	b	e	b	d	e	a
1	2	3	4	5	6	7	8	9	10	11	12

Layer 4: read a two-hop pointer at your two-hop target.

Row i	Stored $f^2(i)$	Read row j	Copy $f^2(j)$	New $f^4(i)$
2	0	0 (null)	0	0
4	0	0 (null)	0	0
8	2	2	0	0
9	4	4	0	0
11	8	8	2	2

$f^4(11) = f^2(8) = 2$. Therefore the four-hop answer is $X_2 = a$.

Appendix D.1, doubling construction (pp. 26–27). Same input and same start position.

The invariant explains all inputs at once

After $2 + t$ layers, every position i stores

$$P_t(i) \;=\; f^{2^t}(i) \quad \text{and} \quad X_{P_t(i)}.$$

Current row | Target row to read | Copy from that row

i	$j = P_t(i)$	$P_t(j)$ and $X_{P_t(j)}$

$$P_{t+1}(i) \;=\; P_t(P_t(i)) \;=\; f^{2^{t+1}}(i)$$

Base case: $t = 0$ is layer 2.
Every row follows this induction; null targets stay null.

Appendix D.1 (pp. 26–27). The invariant below is the power-of-two part of the proof.

Why the construction is causal and narrow

1. Every jump target satisfies $P_t(i) \leq i$: causal attention can read it.
2. A row stores only a constant number of pointer and symbol fields.
3. Each field needs $O(\log N)$ bits, since $|\Sigma| \leq N$.
4. Each new layer doubles the number of composed pointer steps.

Theorem 4.2; Appendices A.1 and D.1 (pp. 6, 13–14, 26–27).

The example isolates the doubling argument

Number of hops	Layers in the construction
1	2
2	3
4	4
8	5
16	6

The theorem covers all k; this proof sketch uses $k = 2^t$.

Theorem 4.2 and Appendix D.1 (pp. 6, 26–27). General-k source bookkeeping is discussed in the appendix notes.

Can a small model use fewer layers?

ASSUMING THE ONE-VERSUS-TWO CYCLE CONJECTURE

Let k be even with $k = \Theta(N^{\xi})$, $0 < \xi \leq 1/2$. Fix $\epsilon \in (0, 1)$.

$$mH = O(k^{1-\epsilon})$$

$$L = \Omega(\log k)$$

For causal models correct on every valid input.

Both growing k and the small mH budget are required.

Corollary 4.3 (p. 6); Appendix D.2. Conditional on Conjecture 2.4.

Conjecture 2.4

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.

Conjecture 2.4 (see, e.g., Ghaffari et al., 2019).

Conjecture 2.4 (p. 4); Appendix D.2. The conjecture is an assumption, not a theorem proved here.

A cycle walk becomes an induction task

1. Encode each graph edge as a short token gadget, producing a sequence X in $O(1)$ MPC rounds.
2. Two induction hops implement one step around the cycle; after k hops, the answer differs for one cycle versus two.
3. Repeat the edge list enough times: $|X| = O(k^2)$, so the Transformer receives the encoded cycle instance.
4. If a shallow Transformer solved hop_k, Theorem 3.4 would turn it into an MPC distinguisher using $O(L)$ rounds.

Appendix D.2 (pp. 27–28). High-level reduction; explicit edge gadgets are in the appendix.

The cycle reduction yields the depth lower bound

5 A Transformer of depth L becomes an MPC distinguisher using $O(L)$ rounds.

Therefore, conditionally, $L = \Omega(\log k)$.

Theorem 3.4, Conjecture 2.4, and Appendix D.2 (pp. 4–5, 27–28).

Can training find a parallel solution?

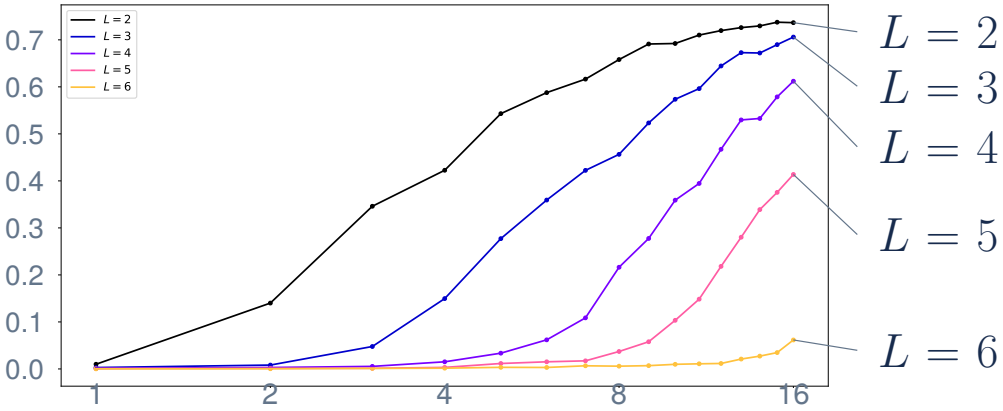
Task / model	Experimental choice
Context and alphabet	$N = 100, \quad \lvert\Sigma\rvert = 4$
Requested hops	$k \in \{0, 1, \ldots, 16\}$, supplied in the input
Depth	$L \in \{2, 3, 4, 5, 6\}$
Main model family	$m = 128, \quad H = 4$

Causal GPT-2; Adam; fresh samples at each training step.

Section 4.2 and Appendix G.1, Tables 1–3 (pp. 6, 37–38).

Each extra layer extends the low-error range

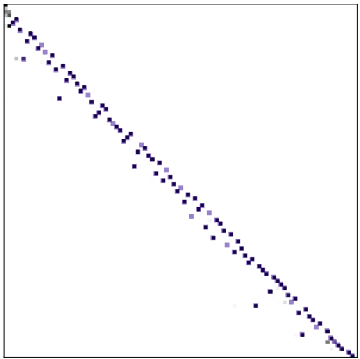
Error on non-null targets versus requested hops k.



Original Figure 6 (p. 39), enlarged from Figure 2. Evaluated on 100 sequences per k; null targets excluded.

How to read an attention panel

Layer 2, head 4; six-layer model; 16-hop task.



A mark links a query position to a position it can read.

Darker marks mean larger learned attention weights.

Colored marks show one-hop targets predicted by f.

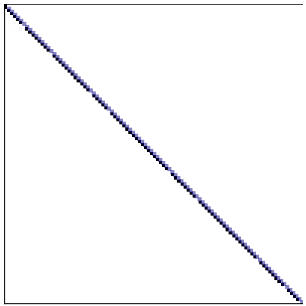
Alignment is evidence, not proof.

Original layer-2/head-4 panel of Figure 10 (p. 43); Appendix G.4 (p. 42). Grayscale weights and the authors' colored overlay are preserved.

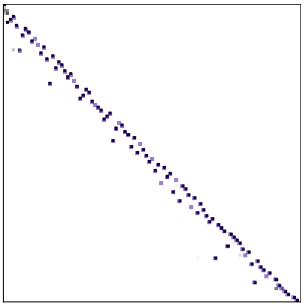
Compare the short and long jumps

One six-layer model, one 16-hop input.

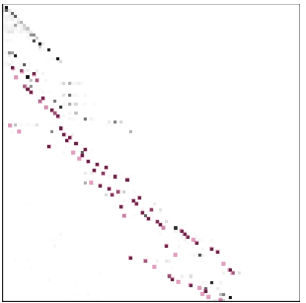
Layer 1: previous



Layer 2: one hop



Layer 6: eight hops



A later head aligns with a longer jump.

Original selected panels of Figure 10 (p. 43); interpretation from Appendix G.4 (p. 42). One model and one input.

Experiments and lower bounds use different regimes

The main experiment has $m = 128 > N = 100$.

The theoretical lower bound requires a small mH relative to growing k.

The data also use a small alphabet and a structured random distribution.

Learning evidence is not a lower-bound proof.

The paper explicitly discusses this width/context mismatch.

Appendix G.2 (p. 39) and G.4 (pp. 42–44).

Why compare modified models?

Other models impose different structures on information flow.

They cannot all exploit parallel communication in the same way as full self-attention.

Section 5 overview (pp. 7–9); Definitions 2.1–2.2 and 5.5. The comparisons change one computational resource at a time.

A different communication model

We now use sequential blackboard communication.

Earlier: MPC	Now: blackboard
All machines send at once. Addressed messages arrive at the end of a round.	Players write in fixed order. The board is public. $P_k, P_{k-1}, \ldots, P_1$.

These reductions use a proved communication theorem.
They do not assume the cycle conjecture.

Definition 2.3 (p. 3) versus Definition E.1 (p. 28). The two notions of a communication round are distinct.

Pointer chasing reveals one address at a time

Player P_j knows a matching $F_j : [w] \to [w]$.

$$v_1 \xrightarrow{F_1} v_2 \xrightarrow{F_2} v_3 \xrightarrow{F_3} \cdots \xrightarrow{F_k} v_{k+1}$$

The starting vertex is known. The task is to identify which half contains the endpoint.

The next queried address is not known in advance.

Appendix E.1, Definitions E.1–E.2 (pp. 28–29). w here denotes the number of vertices per graph layer.

Why the speaking order matters

The start is 1; each lookup table is private.

P_1: private F_1	P_2: private F_2	P_3: private F_3

Our view: $1 \xrightarrow{F_1} 3 \xrightarrow{F_2} 2 \xrightarrow{F_3} 4$

Speaking order: $P_3 \longrightarrow P_2 \longrightarrow P_1$

Pass 1: P_2 speaks before P_1 reveals $F_1(1) = 3$.

It may not yet know that entry 3 of its table is needed.

Definitions E.1–E.2 (pp. 28–29). The values 1, 3, 2, 4 are a teaching example; they do not establish a worst-case lower bound.

Wait for another pass, or send more information

The start is 1; each lookup table is private.

P_1: private F_1	P_2: private F_2	P_3: private F_3

Our view: $1 \xrightarrow{F_1} 3 \xrightarrow{F_2} 2 \xrightarrow{F_3} 4$

Speaking order: $P_3 \longrightarrow P_2 \longrightarrow P_1$

Pass 2: P_2 can reveal $F_2(3) = 2$, but P_3 has already spoken.

Pass 3 can reveal $F_3(2) = 4$.

Larger messages can share more table entries in earlier passes.

Definitions E.1–E.2 (pp. 28–29). The values 1, 3, 2, 4 are a teaching example; they do not establish a worst-case lower bound.

Few passes require large messages

R counts complete reverse-order passes.

B bounds one player's message, in bits.

$$R \geq k \qquad \text{or} \qquad B = \Omega(N/k^6)$$

The input is split into the paper's k player blocks.

Success probability must be at least $2/3$.

Next: identify each model's passes and message budget.

Proposition E.4 (p. 29), obtained from Proposition E.3. The k^5 to k^6 conversion is retained in the appendix.

A recurrent state is the message across a cut



m_{RNN} hidden-state bits

For an L-layer recurrent model solving the final-position task:

$$L \geq k \quad \text{or} \quad m_{\mathrm{RNN}} = \Omega(N/k^6).$$

The paper also covers inference-equivalent state-space examples.

Corollary 5.2 and Appendix E.2 (pp. 7–8, 30). The paper measures recurrent hidden-state width in bits.

A concrete polynomial separation

Choose a power-of-two hop count $k = \Theta(N^{1/10})$.

Recurrent: $L < k$	Transformer
$m_{\mathrm{RNN}} = \Omega(N/k^6)$	$m = O(1)$ coordinates
$= \Omega(N^{2/5})$ bits	$L = O(\log N)$ layers

Transformer: $O(\log N)$ stored bits per row.
Recurrent model: polynomially many state bits.

Algebraic illustration of Theorem 4.2 and Corollary 5.2. Choose a power-of-two $k = \Theta(N^{1/10})$.

A small kernel summary limits communication

$$Q'K'^{\top}V \;=\; Q'\big(K'^{\top}V\big)$$

The shared summary $K'^{\top}V$ has $m' \times m$ entries per head.

Each player can communicate its part using $O(mm'Hp)$ bits.

$$L \;\geq\; k \quad \text{or} \quad mm'Hp \;=\; \Omega(N/k^6).$$

Scope: kernel-factorized attention, not all fast implementations.

Corollary 5.3 and Appendix E.3 (pp. 8, 30–31). m' is the kernel feature dimension.

Local windows and global positions

Query $i = 10$, window radius $w = 1$, global spacing $g = 4$.



Allowed keys: $|i - j| \leq w$ or $j \equiv 0 \pmod{g}$.

$$L \geq k \quad \text{or} \quad \left(w + \frac{N}{gk}\right) mHp = \Omega(N/k^6)$$

Messages carry boundary rows and global rows.

Corollary 5.4 and Appendix E.3 (pp. 8, 31). This is the particular Longformer-type mask formalized in the paper.

Depth and generated tokens are different resources

Let N_{CoT} be the number of generated intermediate tokens.

The same one-layer model repeatedly appends its next output.

$$N_{\mathrm{CoT}} = \Omega(k) \quad \text{or} \quad mHp = \Omega(N/k^6).$$

This lower bound is only for one-layer models.

Deeper chain-of-thought models are not settled here.

Corollary 5.6 (p. 9), stated here asymptotically; Appendix E.4. Scope: a one-layer causal Transformer.

One communication argument, several separations

Architecture	Information passed between input blocks
Recurrent model	A hidden state
Kernel attention	A feature-summary matrix
Local + global mask	Boundary and global embeddings
One-layer CoT	A summary for each new query

Each restriction gives a bounded-message simulation.

Worst-case results for the specific models studied.

Corollaries 5.2–5.4 and 5.6. These lower bounds use communication complexity, not the cycle conjecture.

Why graph edges can be a bottleneck

Message-passing GNN

Messages follow the input graph's edges.

Transformer

Messages can cross arbitrary token pairs.

$$L\sqrt{m} \quad = \quad \widetilde{\Omega}\left(N^{1/4}\right)$$

Some graph instances force this GNN tradeoff.

Scope: this message-passing model, not all graph architectures.

Theorem 5.1 (restating Loukas, 2019) and Section 5.1 (p. 7). $\widetilde{\Omega}$ suppresses logarithmic factors.

What the paper establishes

Can small representations compute it?

Explicit constructions and depth–size lower bounds.

Can training find a useful solution?

Evidence on the synthetic multi-hop task.

Does this prove fast hardware execution?

Hardware runtime is not settled by these bounds.

The model permits arbitrary row-wise computation.

Sections 1 and 6; modeling assumptions in Section 2.1; Appendix G (pp. 1, 3, 9, 36–52).

Takeaways

Transformers and Massively Parallel Computation (MPC) are computationally equivalent

Parallelism is the fundamental computational advantage of transformers

Non-parallelizable and sub-quadratic architectures suffer a severe performance gap

Sections 1 and 6; modeling assumptions in Section 2.1; Appendix G (pp. 1, 3, 9, 36–52).

The notation in one place

Symbol	Meaning
$N,\ m,\ H,\ L$	Context length, width, heads, layers
p	Bits per scalar or word
$q,\ s,\ R$	MPC machines, local words, rounds
γ, δ	Global- and local-memory exponents
$f(i),\ f^k(i)$	One-hop pointer and its repeated composition
$\mathrm{hop}_k(X)_i$	Symbol at the resulting pointer, or null

Definitions 2.1–2.3 and 4.1; Appendix E.1.

How the correct source rows are selected

For each receiver i and destination set S, use short codes u_i, w_S with

$$u_i^\top w_S = 1 \text{ if } i \in S, \qquad u_i^\top w_S \leq \tfrac{1}{2} \text{ otherwise.}$$

Give the dummy key score $3/4$ and value 0.

Select the contacting sources; otherwise select the dummy.

The margin is $1/4$. Apply Lemma A.2.

Proposition B.1 and Appendix F.2 (pp. 14, 33–34). The existence of short codes is imported in that proof.

Every small packet set has a clean slot

Let $A \in \{0,1\}^{n_{\text{addr}} \times d}$ assign packets to slots.

Each of any b packets must be alone in some column.

$$d \quad = \quad O(b^2 \log n_{\text{addr}})$$

Routing uses $n_{\text{addr}} = N^2$ source–destination pairs and $b = s^2$.

Therefore $d \quad = \quad O(s^4 \log N)$.

Fact B.2 (pp. 16–17). Symbols n_{addr} and b rename the address universe and interference bound.

Why the simulation uses several heads

A naive head pads every message to s words.

$$m \quad = \quad O(s^5 \log N)$$

Route different message sizes in separate heads.

$$m = O(s^4 \log N), \qquad H = O(\log \log N).$$

Larger messages mean fewer messages.

Lemma B.7 and Appendix B.2 (pp. 19–20).

Avoid storing huge exponentials

A subtree carries its normalized output S_B and its log-normalizer $\log Z_B$.

$$\log Z = \log\left(\Sigma_B \, e^{\log Z_B}\right)$$

$$S = \Sigma_B \frac{Z_B}{\Sigma_C \, Z_C} S_B$$

The output remains a convex combination of bounded values.

Rounding error is controlled using $p = \Theta(\log N)$.

Appendix C.1, precision analysis (pp. 24–25).

All pointers in the running example

i	1	2	3	4	5	6	7	8	9	10	11	12
X_i	b	a	e	b	c	a	b	e	b	d	e	a
$f(i)$	0	0	0	2	0	3	5	4	8	0	9	7
$f^2(i)$	0	0	0	0	0	0	0	2	4	0	8	5
$f^4(i)$	0	0	0	0	0	0	0	0	0	0	2	0

Examples: $f^2(12) = 5$ returns c; $f^4(11) = 2$ returns a.

Direct calculation from Definition 4.1 on the paper's sequence $baebcabebdea$.

Select an address with a short code

For a target position $\tau \le i$, choose query and key coordinates

$$\begin{aligned} q_i &= \big(\cos(2\pi\tau/N),\ \sin(2\pi\tau/N)\big) \\ k_j &= \big(\cos(2\pi j/N),\ \sin(2\pi j/N)\big) \\ q_i^\top k_j &= \cos\big(2\pi(\tau - j)/N\big)\,. \end{aligned}$$

Among $j \in \{1, \dots, N\}$, the target has score 1 and a gap $\Omega(N^{-2})$.

Lemma D.1 and Appendix F.4 (pp. 26, 35). The nonzero target must be allowed by the causal mask.

Why a score margin matters

Let every losing score be at least $\xi = N^{-O(1)}$ below the maximum.
Scale all scores by a sufficiently large factor c.

$$\frac{\text{total losing exponential weight}}{\text{one maximal exponential weight}} \leq Ne^{-c\xi}.$$

Make this leakage smaller than the output rounding scale.

Softmax is not literally hardmax before rounding.

Lemma A.2 and Appendix F.1 (pp. 13, 32–33). Exactness is in the specified rounded implementation.

The general-k bookkeeping needs care

Theorem 4.2 is stated for all k with $L = \lfloor \log_2 k \rfloor + 2$.

The displayed binary base case reverses the zero/one-bit interpretation.
The subsequent accumulator update also has an index mismatch as printed.

We prove the power-of-two invariant explicitly.

The sharp general-k recurrence is not independently verified here.

Theorem 4.2 and Appendix D.1, printed pp. 26–27. This is a reading caution, not a claimed counterexample to the theorem.

An edge becomes a short token sequence

For an edge $\{u, v\}$, form the nine-token gadget

$$e \;=\; \star\; u \;\; \dagger \;\; v \;\; u \;\; \dagger \;\; v \;\; \star \;\; _.$$

Append a query u: the pointer visits positions $10 \to 6 \to 4$.

$$\mathrm{hop}_2(e \circ u)_{10} \;=\; v.$$

Appending v similarly returns u after two hops.

Appendix D.2 (pp. 27–28). $\star$, $\dagger$, _ are separator symbols distinct from all vertex labels.

Why the exponent becomes k^6

$$R < k \quad \Longrightarrow \quad B = \Omega(w/k^5)$$

The encoded input has $N = \Theta(wk)$ tokens.

$$w = \Theta(N/k)$$

$$B = \Omega\left(\frac{N/k}{k^5}\right) = \Omega(N/k^6).$$

Propositions E.3–E.4 (p. 29). Here w is the matching-layer size and B is a per-player message budget in bits.

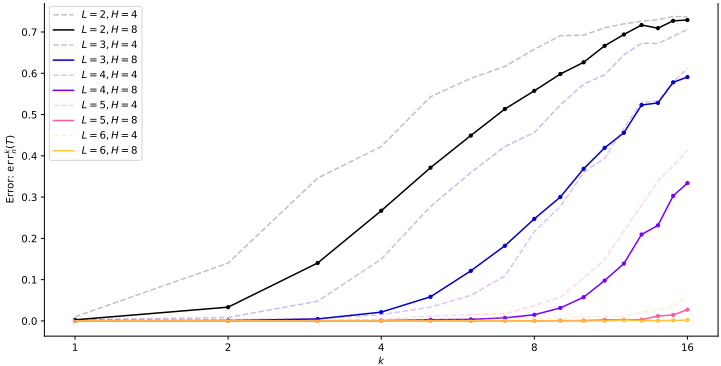
Width in coordinates is not width in bits

Representation	Information held
Transformer: m coordinates, p bits per coordinate	mp bits per position
Paper's recurrent state: m_{RNN} binary entries	m_{RNN} bits
An RNN with d coordinates of p bits each	dp bits

Compare the communication budgets in the same units.

Section 2.1; Corollary 5.2 and its RNN definition (pp. 3, 7).

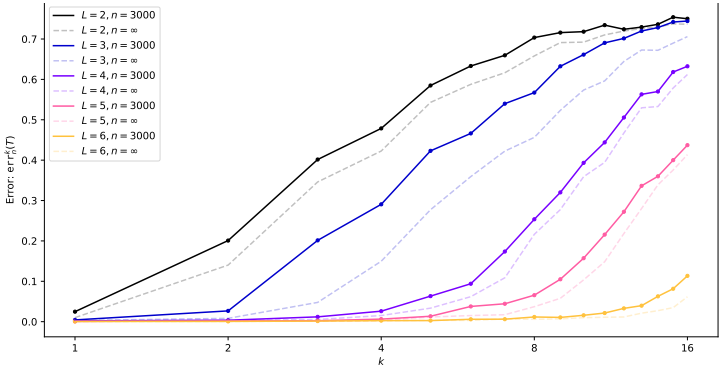
The wider experimental model



Dashed: $(m, H) = (128, 4)$. Solid: $(m, H) = (256, 8)$.

Original Figure 9 and Appendix G.3 (p. 41). The wider family increases both m and H.

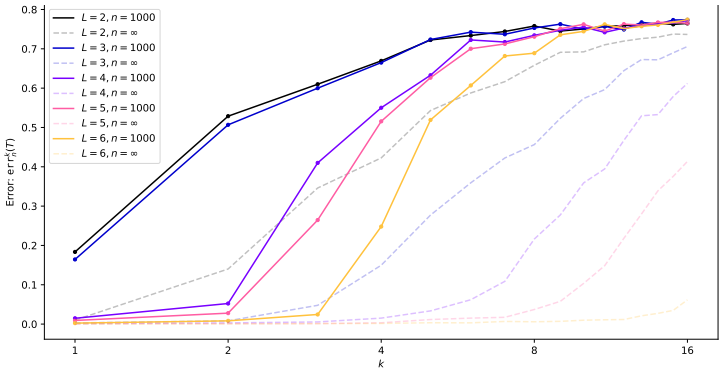
Three thousand fixed examples



Solid: 3,000 fixed examples. Dashed: fresh samples each step.

Original Figure 16 and Appendix G.5 (p. 50). Evaluated on 100 sequences per task.

Less data exposes another depth effect



Solid: 1,000 training examples. Errors are much larger.

Original Figure 17 and Appendix G.5 (pp. 50–51).

Conventions and theorem statements

1. Definition 4.1 permits $j = i$; Lemma D.2 is worded with $j < i$.
2. The one-layer CoT theorem and proof differ by one in counting the final output.
3. The cycle reduction's prose flips the labels of the two output cases.

Use Definition 4.1 and the asymptotic CoT conclusion.

Definition 4.1; Lemma D.2; Corollary 5.6 and Appendix E.4; Appendix D.2 (pp. 6, 9, 26–28, 31–32).

Four parameters describe the model

Symbol	Meaning
N	Number of input positions
m	Coordinates stored at each position
H	Attention heads in each layer
L	Number of layers

The input length N is the main quantity that grows.

Section 2.1, Definitions 2.1–2.2 (p. 3).

Different messages can cross in one round



Receivers need not know their senders in advance.

Teaching example of addressed routing from Definition 2.3. Payload labels stand for short messages.

Why does the width contain a fourth power?

At most s contributing sources, each with at most s packets.

$$s \times s = s^2 \quad \text{potentially interfering packets}$$

A fixed isolating family uses $O(s^4 \log N)$ slots.

Each packet carries β payload words and its metadata.

One layer, one head, $m = O(s^4 \beta \log N)$.

Lemma 3.2 and Fact B.2 (pp. 5, 16–17). The slot count is a worst-case bound.

What a worst-case separation does not say

1. The model must solve the specified task on all inputs, not just typical data.
2. The hard hop instances can use an alphabet growing with N/k.
3. Changing the attention model differs from speeding up the same attention computation.
4. No result here ranks these architectures on all practical tasks.

Section 5 and Proposition E.4 (pp. 7–9, 29); experimental alphabet from Appendix G.1.

A precise first-copy predicate

Let E_t mark clean slots with destination i.

Their recovered pairs are $P_t = (M_t, S_t)$.

$$\mathrm{keep}_t \;=\; E_t \;\wedge\; \neg \vee_{r<t} \Big(E_r \wedge (P_r \;=\; P_t) \Big).$$

Keep a pair only at its first eligible slot.

f is skipped because $P_f = P_a$.

All comparisons use coordinates of one local vector.

Teaching implementation of Appendix B.1's set output (p. 17), under the arbitrary row-wise functions of Section 2.1 (p. 3).

Why a random column can isolate a packet

Let n_{addr} count possible addresses and κ bound $|U|$.

Choose entries independently with probability $1/(\kappa+1)$.

For fixed U and $u \in U$, isolation has probability

$$\frac{1}{\kappa+1}\left(1-\frac{1}{\kappa+1}\right)^{|U|-1} \geq \frac{1}{e(\kappa+1)}.$$

After d columns, failure for this fixed (U,u) has probability

$$\leq \exp\left(-\frac{d}{e(\kappa+1)}\right).$$

Fact B.2 proof (p. 17). κ renames its interference bound b; n_{addr} renames its address count n.

One table can satisfy all the requirements

There are at most $n_{\text{addr}}^{\kappa+1}$ choices of a failure pair (U, u).

The union bound makes the probability of any failure at most

$$n_{\text{addr}}^{\kappa+1} \exp\left(-\frac{d}{e(\kappa+1)}\right) \quad < \quad 1$$

when $d \geq \lceil 12\kappa^2 \ln n_{\text{addr}} \rceil$. Hence a successful table exists.

$$n_{\text{addr}} = N^2, \quad \kappa = s^2 \quad \Longrightarrow \quad d = O(s^4 \log N).$$

Fix that table once. Use it for every valid routing input.

Fact B.2 proof and its substitution into Lemma 3.2 (pp. 16–17). Asymptotic nondegenerate regime $n_{\text{addr}} \geq 2$.

Sources and further details

Clayton Sanford, Daniel Hsu, Matus Telgarsky

Transformers, Parallel Computation, and Logarithmic Depth

ICML 2024, Proceedings of Machine Learning Research 235.

MPC routing: Appendix B. Reverse simulation: Appendix C.

Hop construction: Appendix D. Lower bounds: Appendix E.

Attention primitives: Appendix F. Experiments: Appendix G.

Previous-paper comparison: Pérez, Barceló and Marinkovic, JMLR 2021.

All theorem numbers, appendix labels and figure crops refer to the supplied sanford24a.pdf, not a later revision.

Local means: already in the same vector

Both numbers are already stored in one vector.



$$g(a, b) = a + b, \qquad g(2, 8) = 10$$

g is an example of a row-wise function.

No information from another position is needed for this addition.

Section 2.1, modeling assumptions (p. 3). Addition is an original teaching illustration.

Another position must communicate its data

Now the numbers are at different positions, with $j < i$.

Position i stores 2

Position j stores 8

A function of the vector at i cannot inspect the vector at j.

Attention must convey the missing value before local addition.

Changing 8 at j to 80 cannot change a purely local result at i.

Definitions 2.1–2.2 and modeling assumptions (p. 3). Original dependency illustration.