Capabilities and Limitations of Transformers in sequential reasoning

Bingbin Liu

Carnegie Mellon University → Simons → Kempner (Harvard)

Jordan T. Ash

Surbhi Goel

Akshay Krishnamurthy

Yuchen Li

Andrej Risteski

Kaiyue Wen

Cyril Zhang

This talk

0. Formalizing reasoning.

Finite-state automata

1. (Theoretical) Capabilities – Shallow solutions to sequential tasks.

Tools from Krohn-Rhodes theory and formal languages.

2. (Practical) Limitations – Imperfect out-of-distribution performance.

Causes and mitigations.

Sequential reasoning tasks

$$
\begin{array}{r}
1 \\
\times\ (-1) \\
\times\ (-1) \\
\times 1 \\
\cdots
\end{array}
$$

arithmetic

Bobo is a corgi.
A corgi is a dog.
A dog is a mammal.
Is Bobo a mammal?

multi-hop reasoning
(polysyllogism)

```
x = 1
for _ in range(10):
  x = x**2 + 1
print(x)
```

program

Universal presence in diverse forms.

Formalizing sequential reasoning

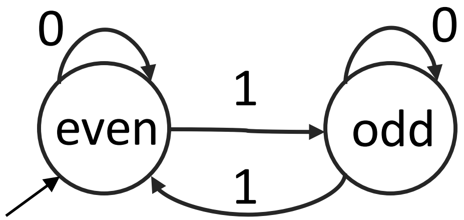
$$\begin{array}{r} 1 \\ \times\ (-1) \\ \times\ (-1) \\ \times 1 \\ \cdots \end{array}$$

arithmetic

$$1 = (-1)^{0}, \quad -1 = (-1)^{1},$$

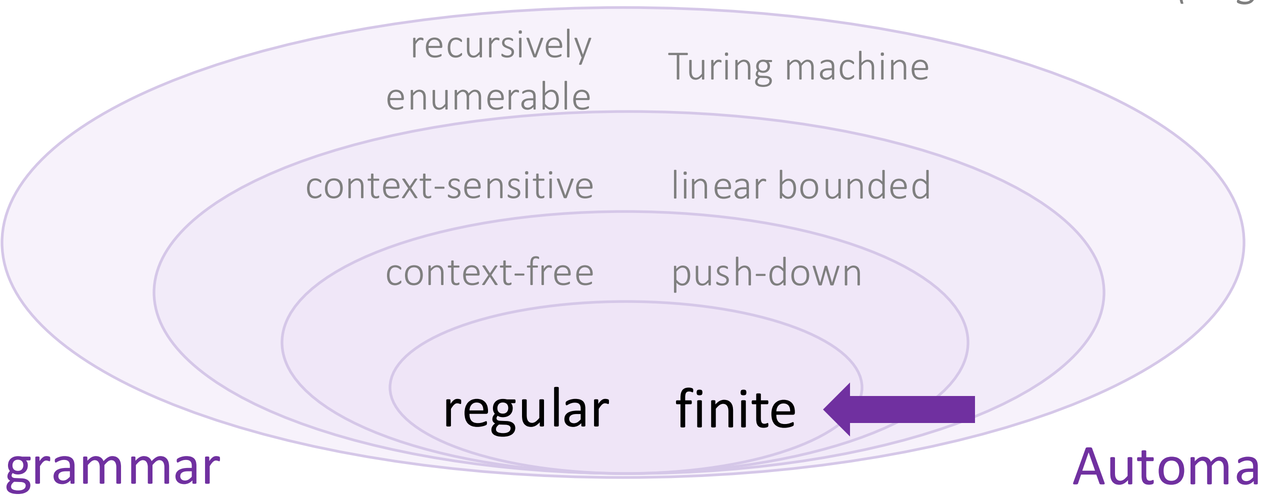
for the exponents:

aka. **parity counter**



Finite-state automata

(regular languages)



Formal grammar

Automata

Sequential reasoning via automata

$$\mathcal{A} = (Q, \Sigma, \delta)$$

states **inputs** **transitions**

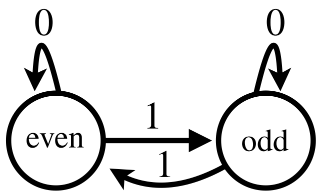
$$q_t = \delta(q_{t-1}, \sigma_t)$$

(Q is finite)

parity counter

$Q = \{\text{even}, \text{odd}\}$

$\Sigma = \{0, 1\}$

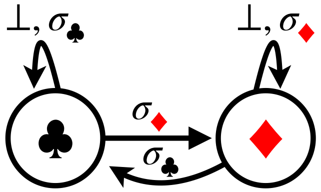


1-bit memory unit

$Q = \{♣, ♦\}$

$\Sigma = \{\sigma_♣, \sigma_♦, \bot\}$

(no-op)



Reasoning = simulating the dynamics of $\mathcal{A}$.

Task: Simulating automata

$\mathcal{A} = (Q, \Sigma, \delta)$

states, inputs, transitions

Simulating $\mathcal{A}$: learn a *seq2seq function* for sequence length T.

Output: q_1 q_2 $\cdots$ q_T $\subset Q^T$ (states)

Model (Transformer/RNN)

Input: σ_1 σ_2 $\cdots$ σ_T $\subset \Sigma^T$ (alphabet)

The Transformer layer

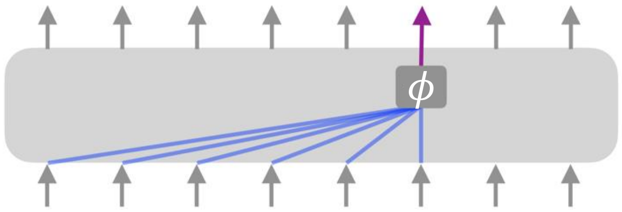
Computation *parallel* across positions.

attention scores: $\sum_j \alpha_{ij} = 1$

$$l^{th} \text{ layer, position } i \in [T]\text{: } x_i^{(l)} = \phi\left(\textstyle\sum_{j \le i} \alpha_{ij}^{(l)}\ x_j^{(l-1)}\right)$$

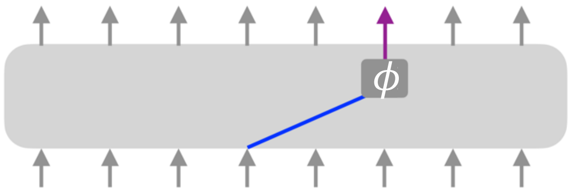
parameters

1. uniform attention / $\overrightarrow{\alpha_i} = [\frac{1}{\mathrm{T}}, \frac{1}{\mathrm{T}}, \cdots, \frac{1}{T}]$



e.g. average, sum.

2. sparse attention / $\overrightarrow{\alpha_i} = [0, \cdots 1, 0, \cdots]$



e.g. selection.

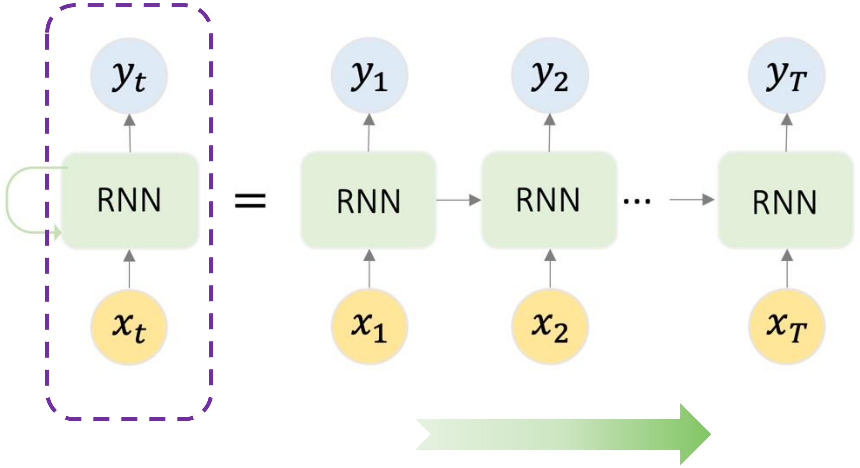
Architecture choices

L (#layers) $\ll T$ (# positions)

Recurrent Neural Nets (RNNs)

sequential across positions

Natural for $q_t = \delta(q_{t-1}, \sigma_t)$

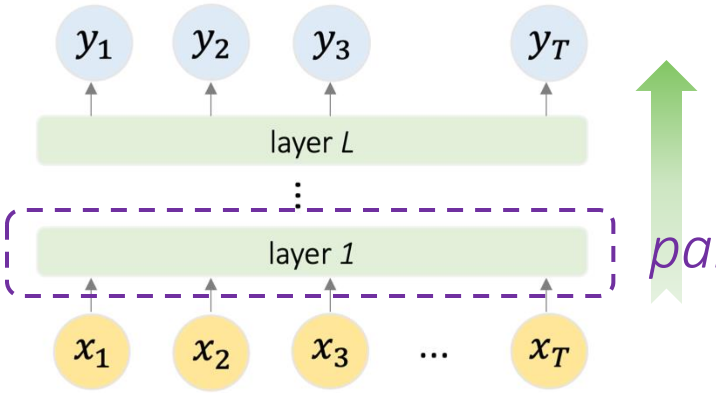


T positions

Transformer

parallel across positions

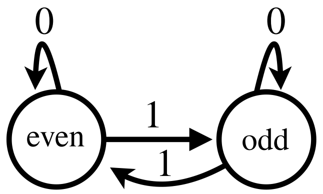
sequential across layers



L layers

parallel

A <u>parallel</u> model for a <u>sequential</u> task?



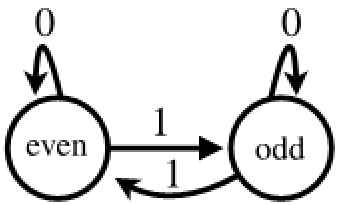
parity

Different ways to simulate automata

$\mathcal{A} = (Q, \Sigma, \delta)$

states, inputs, transitions

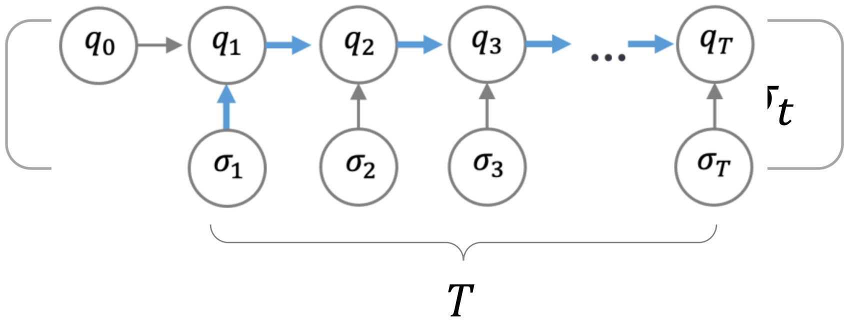
Simulating = mapping from $(\sigma_1, \sigma_2, \cdots, \sigma_T) \subset \Sigma^T$ to $(q_1, q_2, \cdots, q_T) \subset Q^T$.

e.g. parity 

Shortcut

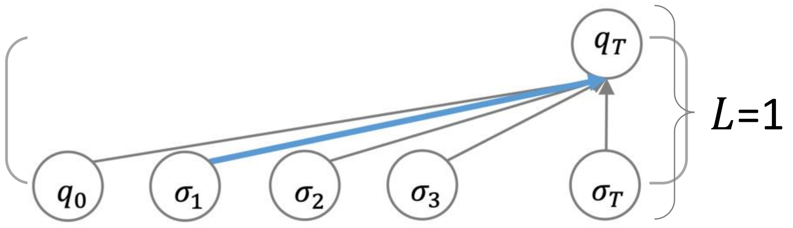
$o(T)$ # sequential steps

Iterative solution



"RNN solutions"

Parallel solution



"Transformer solutions"

Solutions of Reasoning

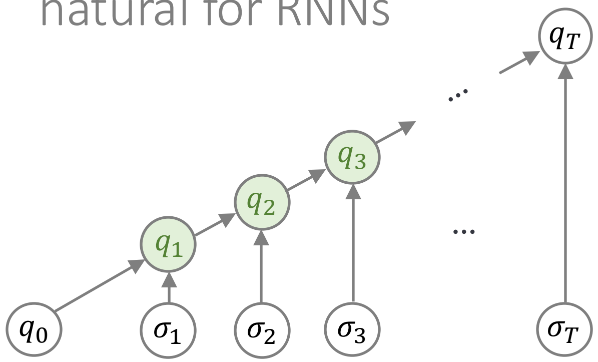
 # steps = # sequential computation steps

$\mathcal{A} = (Q, \Sigma, \delta), \quad q_t = \delta(q_{t-1}, \sigma_t).$

Sequential solutions

(# steps = T)

By δ's definition;
natural for RNNs



iterative state emulation 🐢

Shortcuts *(#steps = $o(T)$)*

Transformer can simulate $\mathcal{A}$ with:

(Thm 1) $\boldsymbol{O(\log T)}$ layers

(Thm 2) $\boldsymbol{\tilde{O}(|Q|^2)}$ layers

Task structure?

Why Transformer?

$O(\log T)$ layers

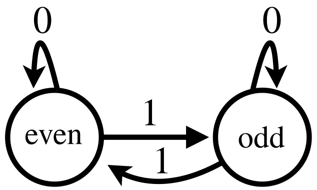
$\mathcal{A} = (Q, \Sigma, \delta)$,
$q_t = \delta(q_{t-1}, \sigma_t)$.

Goal: compute $q_t = \left(\delta(\cdot, \sigma_t) \circ \cdots \circ \delta(\cdot, \sigma_1)\right)(q_0), t \in [T]$.

$\delta(\cdot, \sigma)\colon Q \to Q$

function ⟷ matrix

composition ⟷ multiplication



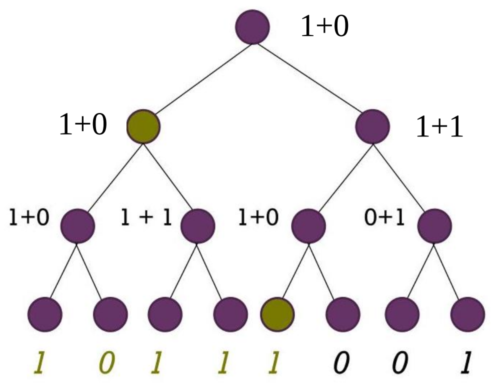
$Q = \{\text{even}, \text{odd}\}$
$\Sigma = \{0, 1\}$

parity counter

$$\delta(\cdot, 0) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},$$

$$\delta(\cdot, 1) = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

associativity



$O(\log_2 T)$

depth-width tradeoff

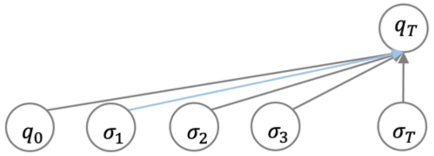
How to get $o(\log T)$ layers?

$$q_t = \left(\delta(\cdot,\sigma_t) \circ \cdots \circ \delta(\cdot,\sigma_1)\right)(q_0)$$

We already have positive results.

- Parity: only need to count #1s.

$$q_t = \left(\sum_{\tau \le t} \sigma_\tau\right) \bmod 2$$



$$f \circ g = g \circ f$$

Counting works for commutative function composition: $O(1)$ layers.

$$f \circ g \neq g \circ f$$

How about *non-commutative* compositions?

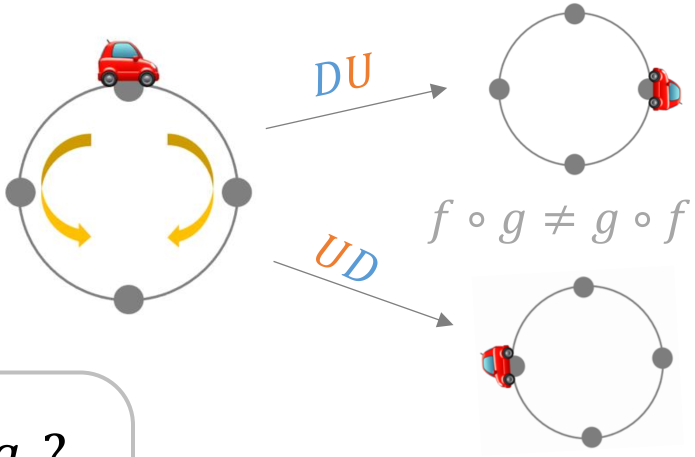
Decomposition

Decomposition: example

$$\mathcal{A} = (Q, \Sigma, \delta), \quad q_t = \delta(q_{t-1}, \sigma_t).$$

$Q = \{$🚗, 🚘$\} \times \{0,1,2,3\}$, $\Sigma = \{D(\text{drive}), U(\text{U–turn})\}$.

1, -1



	q_0	D	D	D	U	D	D	U	U	D	$\rightarrow$	q_t?
Parity:	1	1	1	1	-1	-1	-1	1	-1	-1	$\rightarrow$	🚘
Signed sum:	0	1	1	1	0	-1	-1	0	0	-1	$\rightarrow$	0

$O(1)$ layer each

1. Direction = parity (sum) of U. (parity: {1, -1} ↔ {0, 1})
2. Position = signed sum mod 4 : sign = parity of U.

Decomposition: general

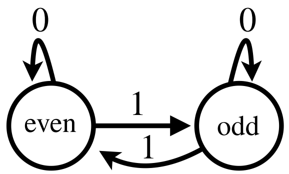
What are we decomposing?

Transformation semigroup: $\mathcal{T}(\mathcal{A}) \coloneqq \{\delta(\cdot, \sigma) : \sigma \in \Sigma\}$ under composition.

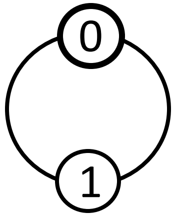
A generalization of group, satisfying only associativity.

parity counter

$Q = \{\text{even}, \text{odd}\}$
$\Sigma = \{0, 1\}$



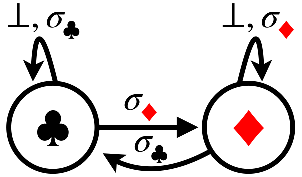
$$\delta(\cdot, 0) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad \delta(\cdot, 1) = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \xrightarrow{\mathcal{T}(\mathcal{A})}$$



cyclic group C_2

memory unit

$Q = \{\clubsuit, \blacklozenge\}$
$\Sigma = \{\sigma_\clubsuit, \sigma_\blacklozenge, \bot\}$

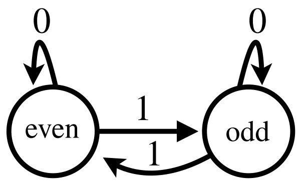


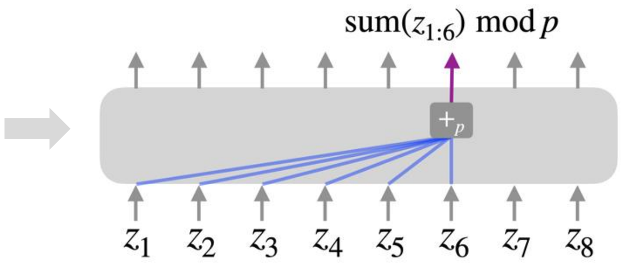
$$\delta(\cdot, \sigma_\clubsuit) = \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix}$$ *... singular → semigroup*

Decomposition: $\tilde{O}(|Q|^2)$ layers

$\mathcal{A} = (Q, \Sigma, \delta)$,
$q_t = \delta(q_{t-1}, \sigma_t)$.

For a subset of $\mathcal{A}$, its $\mathcal{T}(\mathcal{A})$ can be *decomposed* into 2 base factors [Krohn-Rhodes]:
(solvable)

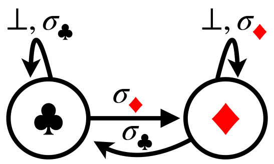


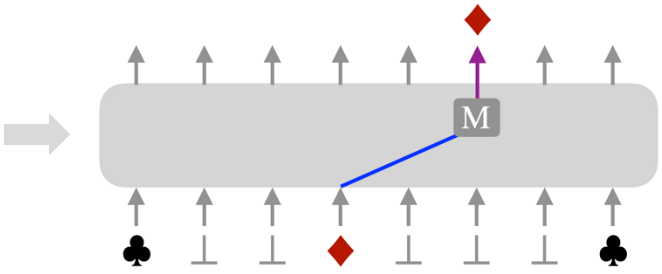


$\delta(q, \sigma) = q + \sigma \bmod p$

mod counter

uniform attention





$\delta(q, \sigma) = \sigma$,
$\delta(q, \perp) = q$.

memory unit

sparse attention

$\tilde{O}(|Q|^2)$ layers
- *Why Transformer*: Each factor representable by 1 Transformer layer.
- Number of factors is $\tilde{O}(|Q|^2)$.

Solutions of Reasoning

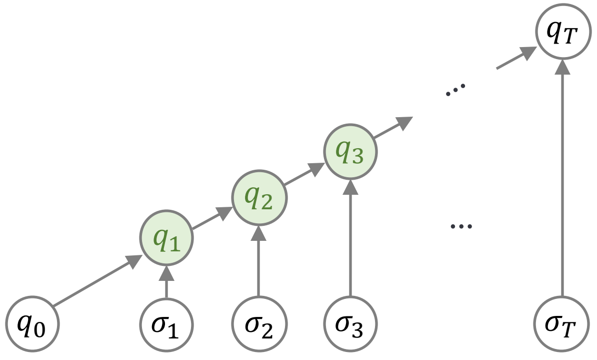
 # steps = # sequential computation steps

$$\mathcal{A} = (Q, \Sigma, \delta), \quad q_t = \delta(q_{t-1}, \sigma_t).$$

Sequential solutions

#steps = T,

by δ or RNNs.



iterative state emulation 🐢

Shortcuts (*#steps = $o(T)$*)

Transformer can simulate $\mathcal{A}$ with:

(Thm 1) $\boldsymbol{O(\log T)}$ layers.

associativity

tree: divide and conquer

(Thm 2) $\tilde{\boldsymbol{O}}(|\boldsymbol{Q}|^2)$ layers.

algebraic structure

Krohn-Rhodes decomposition

(solvable $\mathcal{A}$ only)

Remarks

All $\mathcal{A}$: $\boldsymbol{O(\log T)}$ layers.

Solvable $\mathcal{A}$: $\boldsymbol{\tilde{O}(|Q|^2)}$ layers.

1. Can we improve $O(\log T)$ in general? Likely not.
   - Constant-depth Transformers $\subset TC^0$ [Merrill et al. 21, Li et al. 24; survey by Strobl et al. 23].
   - Some automata are NC^1 complete (e.g. S_5).

   $\rightarrow \Omega(\log T)$ unless $TC^0 = NC^1$.

2. What is special about Transformers?
   - **Parameter sharing**: T times more efficient in size than a circuit.
   - **Parallelism**: can be even shallower than Krohn-Rhodes.
     - *$O(1)$-layer for all abelian groups and a special non-abelian group.*

Representational results → practical insights?

What solutions are found in practice?

Transformers can simulate automata in practice

19 automata, across various depths.

- Good in-distribution accuracy.
- Deeper factorization → more layers.
  - Rows ordered by #factorization steps.

Constructions ≠ empirical solutions

There are multiple constructions.

Infinitely many

- $O(\log T)$ for all $\mathcal{A}$; $\tilde{O}(|Q|^2)$ if solvable.

non-solvable

Transformer depth $\boldsymbol{L}$ (T=100)

automaton	1	2	3	4	5	6	7	8	12	16
Dyck	99.3	100	100	100	100	100	100	100	100	100
Grid_9	92.2	100	100	100	100	100	100	100	100	100
C_2	77.6	99.8	99.9	100	100	99.5	100	99.7	100	100
C_3	54.6	94.6	96.7	99.4	100	100	99.8	100	100	100
C_2^3	65.0	77.9	99.9	97.9	100	99.8	98.2	99.9	95.9	80.6
D_6	25.4	27.2	47.4	75.2	100	100	100	100	100	100
D_8	45.6	98.0	100	100	100	100	100	100	100	100
Q_8	31.6	49.2	59.6	60.4	73.5	99.3	100	100	100	100
A_5	12.5	23.1	32.5	46.7	71.2	98.8	100	100	100	100
S_5	7.9	11.8	14.6	19.7	26.0	28.4	32.8	51.8	97.2	99.9

Infinitely many solutions to Dyck

Dyck language*:* balanced parentheses → capturing *hierarchical* structures.

valid ([ ])

invalid ([ )]

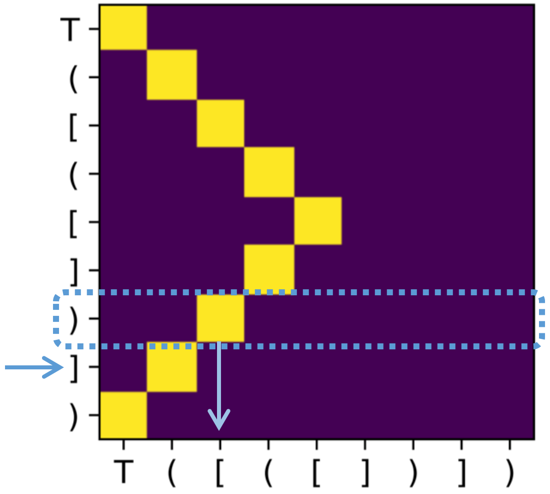
```
for (cond) {
  x[i] = …
}
```

The puppy (which my friend (who lives in NYC) adopted) is fluffy.

Processed by a **stack** or a **2-layer** Transformer.

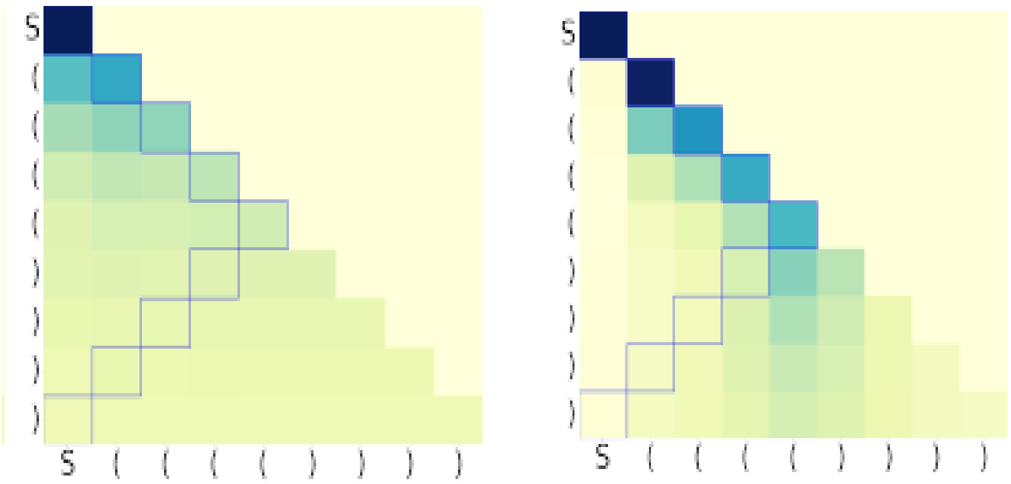
[Ebrahimi et al. 20, Yao et al. 21]

e.g. visualizing 2nd layer attention patterns.



Stack-like

[Ebrahimi et al. 20]



Non-stack-like (more often)

Infinitely many solutions to Dyck

Dyck language*:* balanced parentheses → capturing *hierarchical* structures.

valid ([ ])

invalid ([ )]

Infinitely many solutions, even with a *constrained* 1st layer (i.e. output depending only on type and depth).

[WLLR 23]: all 2-layer Transformers solving Dyck suffice and need to satisfy a *balanced condition*.

~ *a Transformer's version of the pumping lemma.*

(informal: $xyz \in L \rightarrow xy^*z \in L$.)

Processed by a **stack** or a **2-layer** Transformer.

[Ebrahimi et al. 20, Yao et al. 21]

e.g. visualizing 2nd layer attention patterns.

Attention maps may not reflect the task structure.

- Including *non-hierarchical* patterns, e.g. uniform attn.

Infinitely many solutions to Dyck

Dyck language: balanced parentheses → capturing *hierarchical* structures.

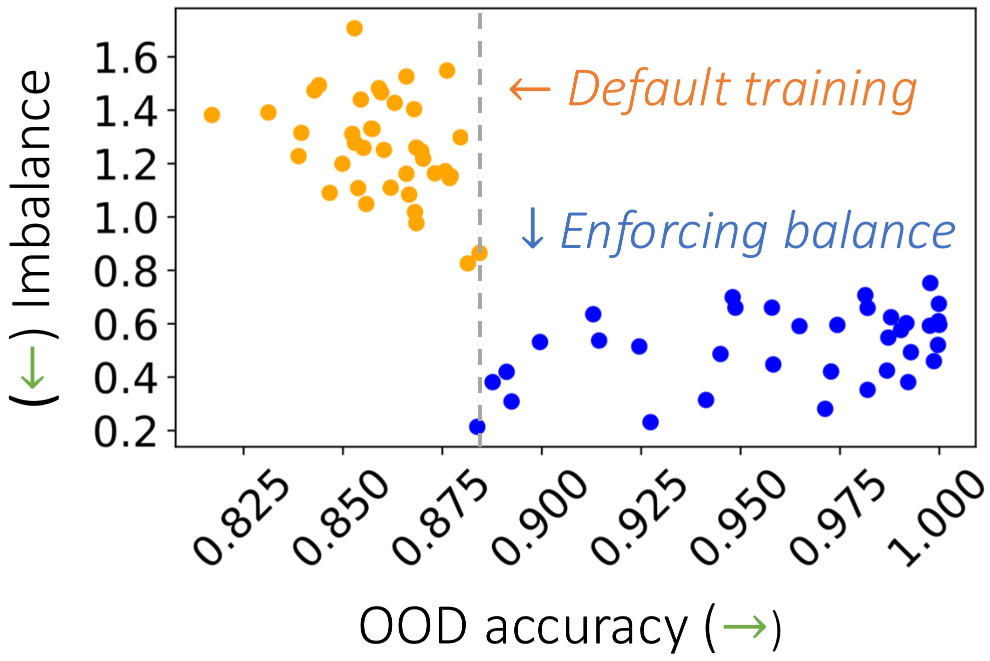
valid ([ ])
invalid ([ )]

Infinitely many solutions, even with a *constrained* 1st layer (i.e. output depending only on type and depth).

- *The balanced condition as a regularizer.*

Processed by a stack or a **2-layer** Transformer.
[Ebrahimi et al. 20, Yao et al. 21]

e.g. visualizing 2nd layer attention patterns.



Representational results → practical insights?

- Constructions ≠ Practical solutions.
  [WLLR23] Infinitely many solutions even for a 2-layer model on Dyck.

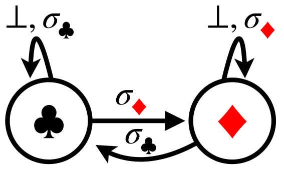
- Why does Transformer struggle OOD? [LAGKZ23]

A simple(st) language based on the memory unit

Recall: one (of the 2) *base factor* of automata decomposition.

1-bit memory unit

$Q = \{0,1\}, \Sigma = \{\sigma_0, \sigma_1, \perp\}$



"flip-flop" automata

Flip-Flop Language (FFL): sequences of instruction-value pairs.

- 3 instructions: `w` (write), `i` (ignore) , `r` (read).
- 2 values: {0, 1} – ***Constraint***: the value for `r` must be the same as the last `w`.

`w 1 i 0 i 1 r 1 w 0 i 1 r 1`

Flip-Flop Language Modeling (FFLM)

Flip-Flop Language (FFL): instruction-value pairs; `r` recalls the most recent `w`.

`w 1 i 0 i 1 r 1 w 0 i 1 r 1`

Task: supervise & evaluate only on the values following `r`.

- Deterministic task; training signals not "drawn" by irrelevant tokens.

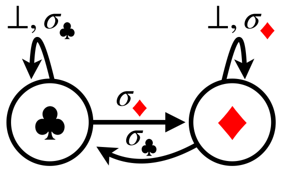
Data distribution: FFL(p_i), where p_i can vary across train/test.

- $p_w = p_r = (1 - p_i)/2$, $p_0 = p_1 = 0.5$. Fix length $T = 512$.

Why Flip-Flop?

1-bit memory unit

$Q = \{0,1\}, \Sigma = \{\sigma_0, \sigma_1, \perp\}$



Flip-Flop Language (FFL)

`w 1 i 0 r 1 w 0 i 1 r 1`

An atomic unit embedded in many reasoning tasks (e.g. automata).

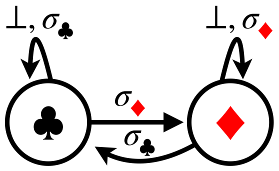
- (1-hop) Induction head [Olsson et al. 22]

`w 1 i 0 r 1 w 0 i 1 r 1`

Why Flip-Flop?

1-bit memory unit

$Q = \{0,1\}, \Sigma = \{\sigma_0, \sigma_1, \perp\}$



Flip-Flop Language (FFL)

`w 1 i 0 r 1 w 0 i 1 r 1`

An atomic unit embedded in many reasoning tasks (e.g. automata).

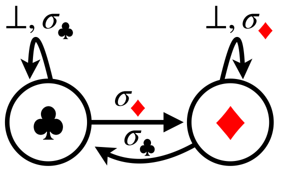
- (1-hop) Induction head [Olsson et al. 22]
- Long-range dependency

`w 1 i 0 r 1 . . . i 1 r`

Why Flip-Flop?

1-bit memory unit

$Q = \{0,1\}, \Sigma = \{\sigma_0, \sigma_1, \perp\}$



Flip-Flop Language (FFL)

`w 1 i 0 r 1 w 0 i 1 r 1`

An atomic unit embedded in many reasoning tasks (e.g. automata).

- (1-hop) Induction head [Olsson et al. 22]
- Long-range dependency
- Closed-domain hallucination [Dziri et al. 22, OpenAI 23]

`w 1 i 0 r 1 w 0 i 1 r 1`

Irrelevant context

[Shi et al. 23]

```
Alice put the keys
on the table.
Bob came in later.
….
Bob left and took
the keys from    .
```

Updated semantics

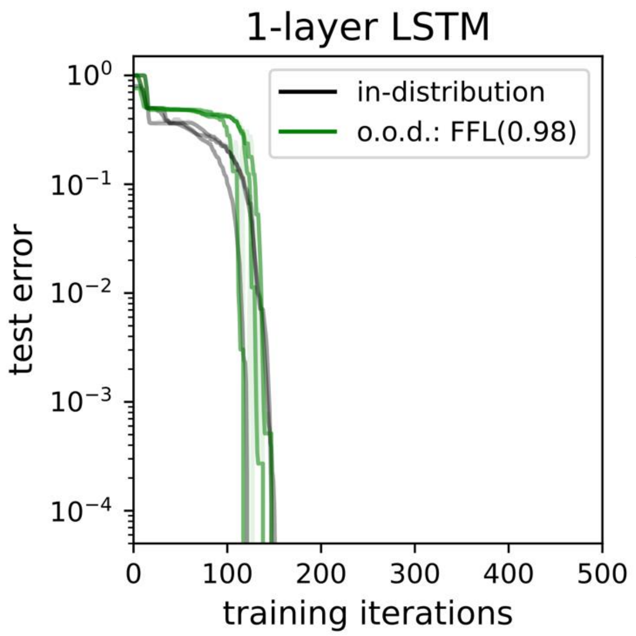
[Miceli-Barone et al. 23]

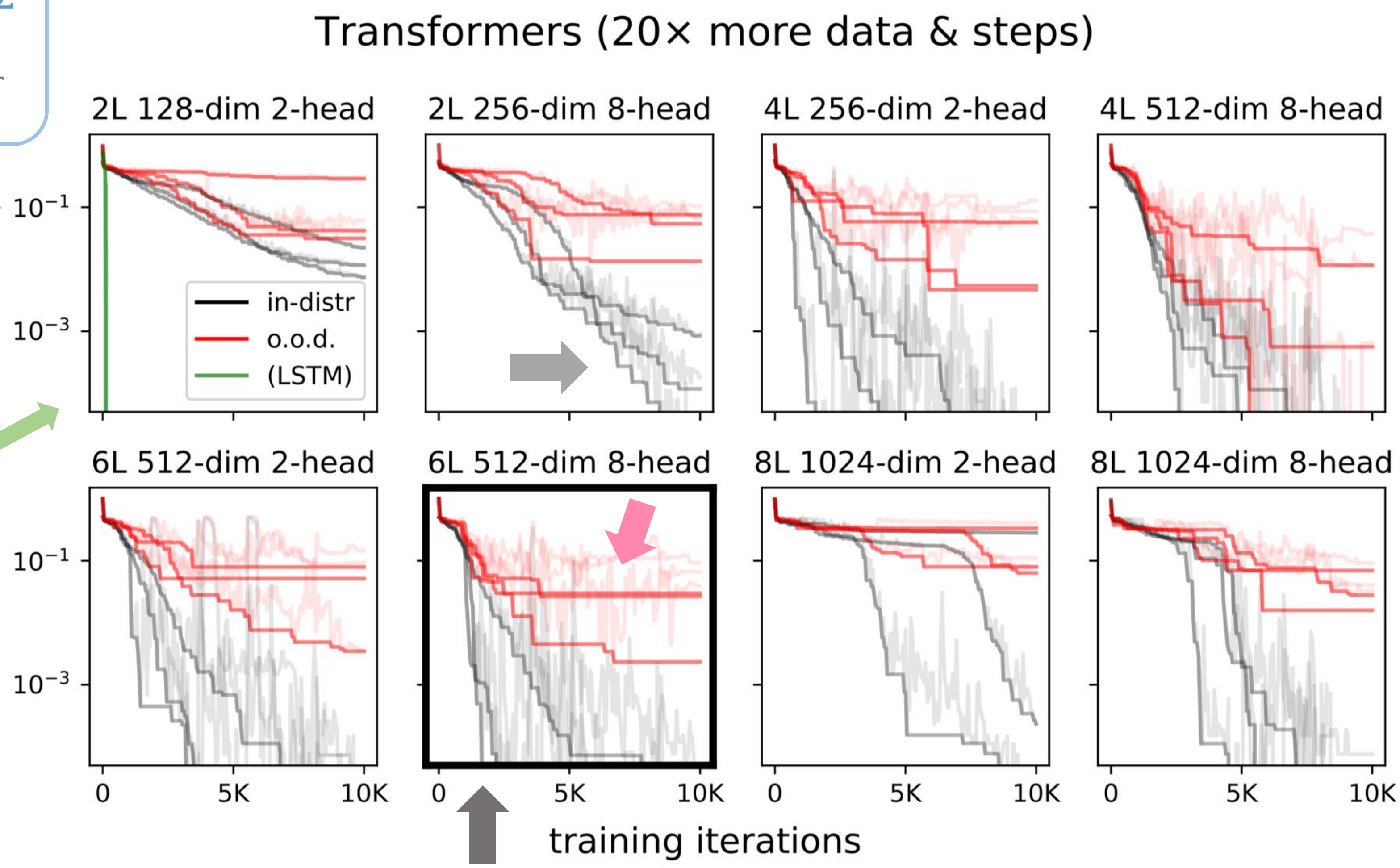
```
def f():
  sum = len
  . . .
  x = [1,2,3]
  . . .
  assert(sum(x))==3
```

Transformer for FFLM

Attention glitches

Train: FFL(0.8) $T = 512$
Test: FFL(0.98) ... *sparser*



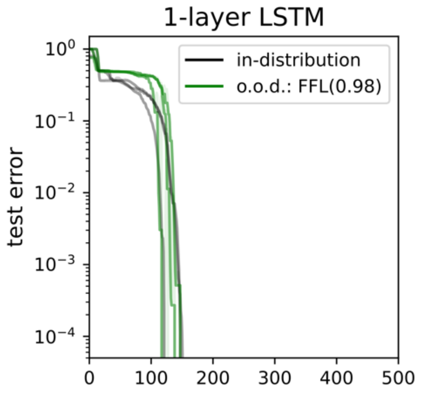


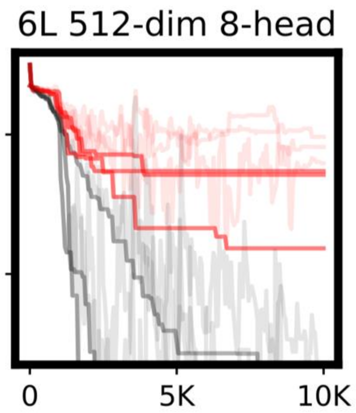
Attention Glitches

FFL(p_i): $p_w = p_r = (1 - p_i)/2$.

Def: *imperfect hard retrieval*.

- Transformers exhibit a long tail of errors.
- 1-layer LSTMs extrapolate *perfectly*.
- Even commercial models are not robust.





User: Hi, let's play a game. There are 3 instructions: "write", "read", and "ignore".

. . .

For example, ...

Now, please answer the following sequence: …

GPT 4o: 50% acc

1, 0, 0, 1, 1, 0, 0, 1, 1, 0

GPT o1-mini: 100% acc

Cause of attention glitches?

FFL(p_i): $p_w = p_r = (1 - p_i)/2$.

Not due to representation power: *2-layer 1-head* suffices (Bietti et al. 23, Sanford et al. 24).

2 potential causes, each related to 1 type of OOD error.

Diluted soft attention: caused by more items (e.g. denser `w`) in the softmax.

$$a_{\max} = \frac{\exp(z_{\max})}{\underbrace{\exp(z_1) + \cdots + \exp(z_t)}_{\textit{to be ignored}} + \exp(z_{max})}$$

- Also identified in prior work [Hahn 20, Chiang & Cholak 22].
- Possible mitigation: Switching to hard attention.

Cause of attention glitches?

FFL(p_i): $p_w = p_r = (1 - p_i)/2$.

Not due to representation power: *2-layer 1-head* suffices (Bietti et al. 23, Sanford et al. 24).

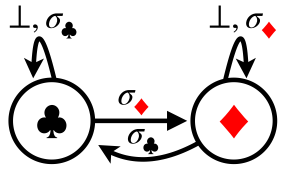
2 potential causes, each related to 1 type of OOD error.

Diluted soft attention: caused by more items (e.g. denser `w`) in the softmax.

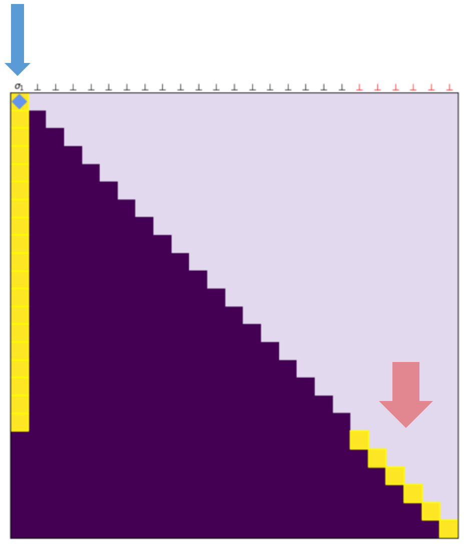
Position over content: lead to wrong argmax.
(e.g. sparser `w`, length gen)

1-bit memory unit

$Q = \{0,1\}, \Sigma = \{\sigma_0, \sigma_1, \perp\}$



Experiments: 1-layer, 1-head models.



Mitigating attention glitches

Data & scale

- Incorporating OOD data.

  Performance ceiling; a few samples can help.
  e.g. "priming" [Jelassi et al. 23]

- Resource scaling: larger, train for longer.

  Fresh samples → better coverage.

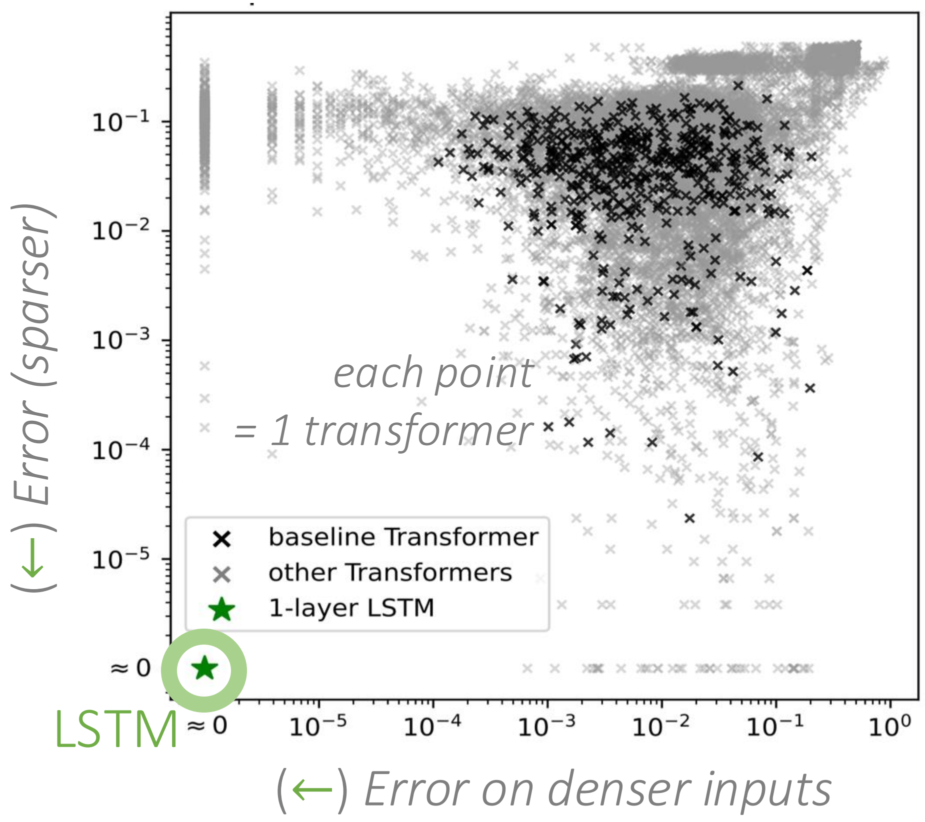
Algorithmic control

- Regularization

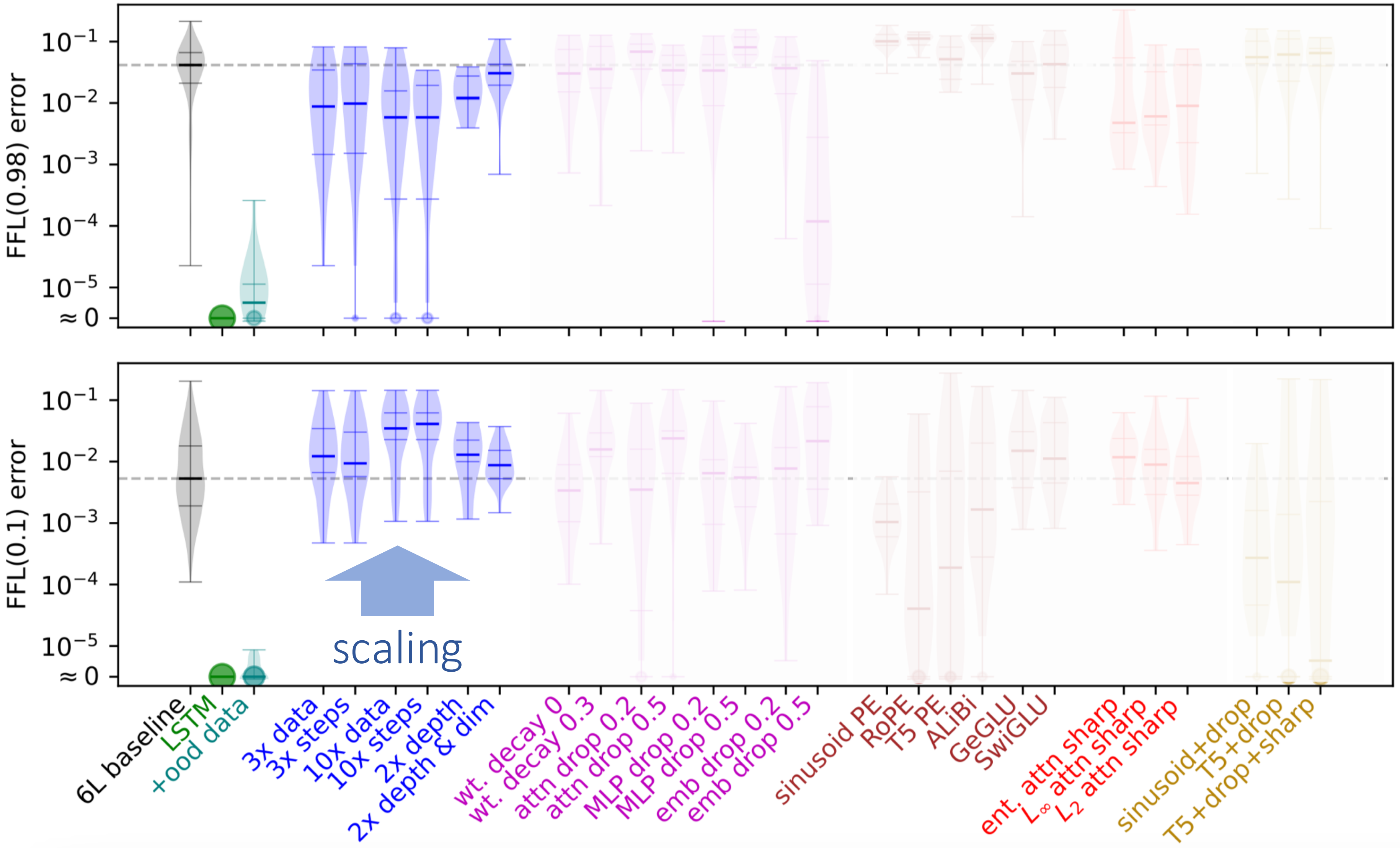
  weight decay, dropout, attention sparsity.

- Architectural choices

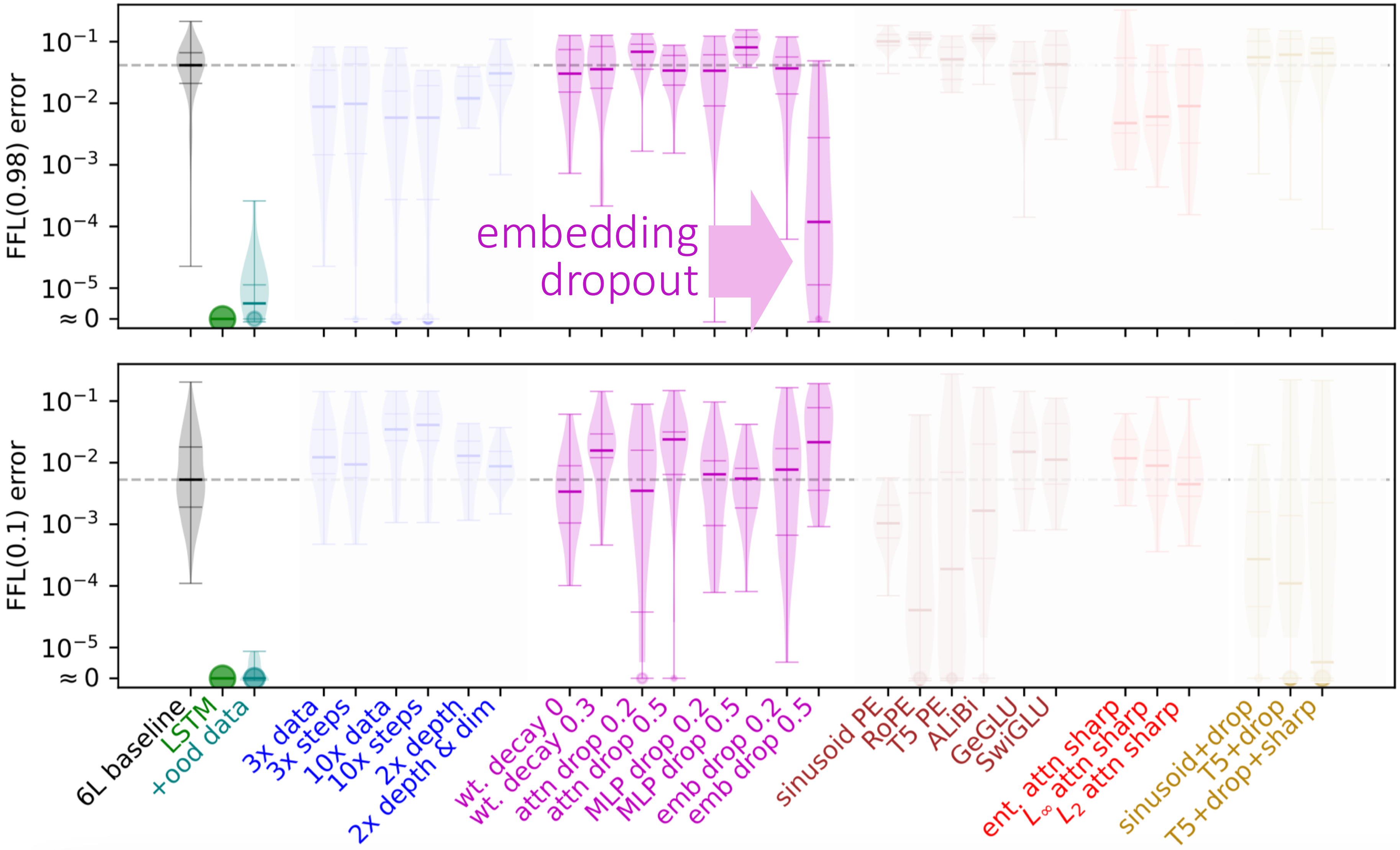
  position encoding, activation.



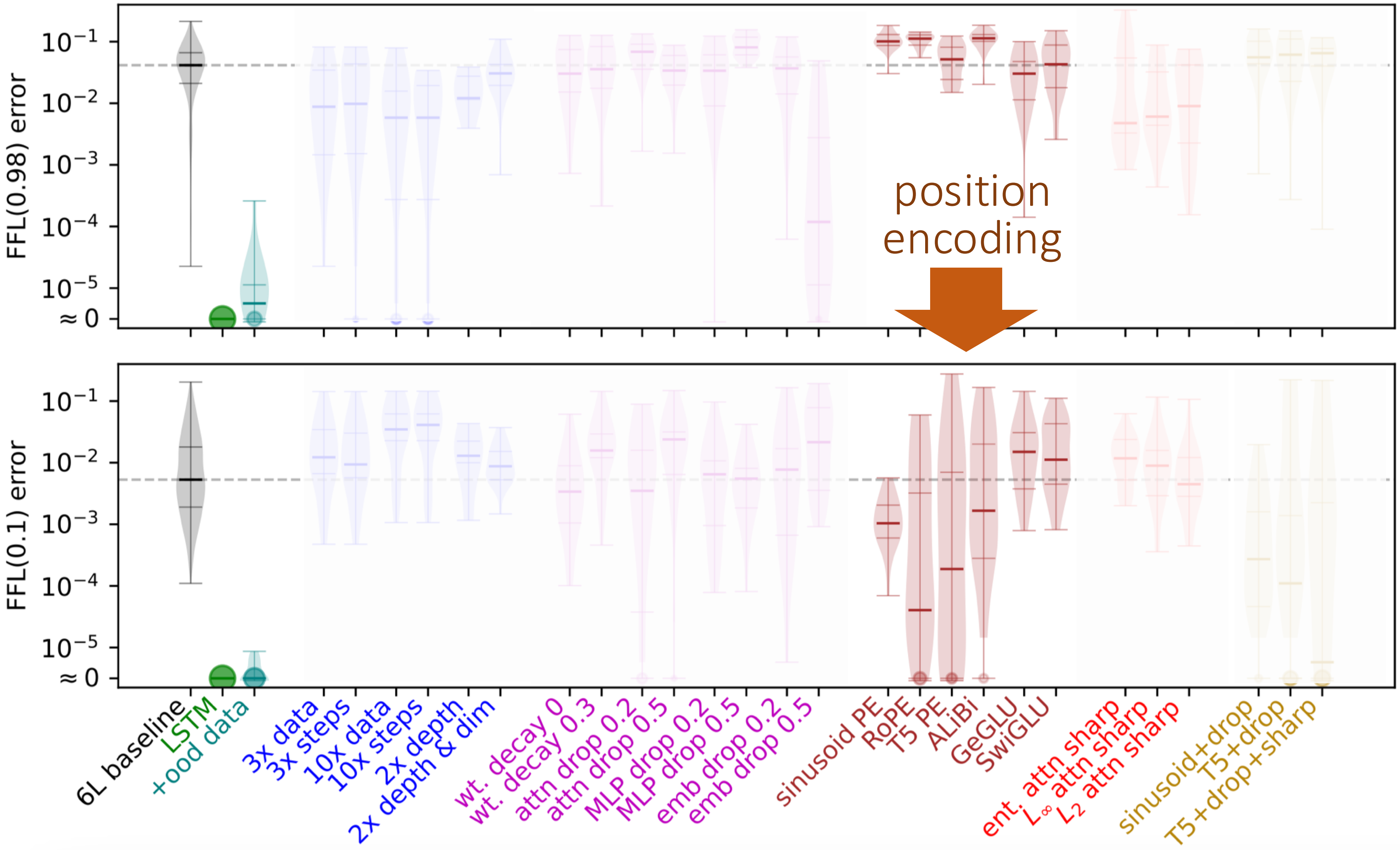
Surprisingly hard to fix: no mitigation helps with *both*



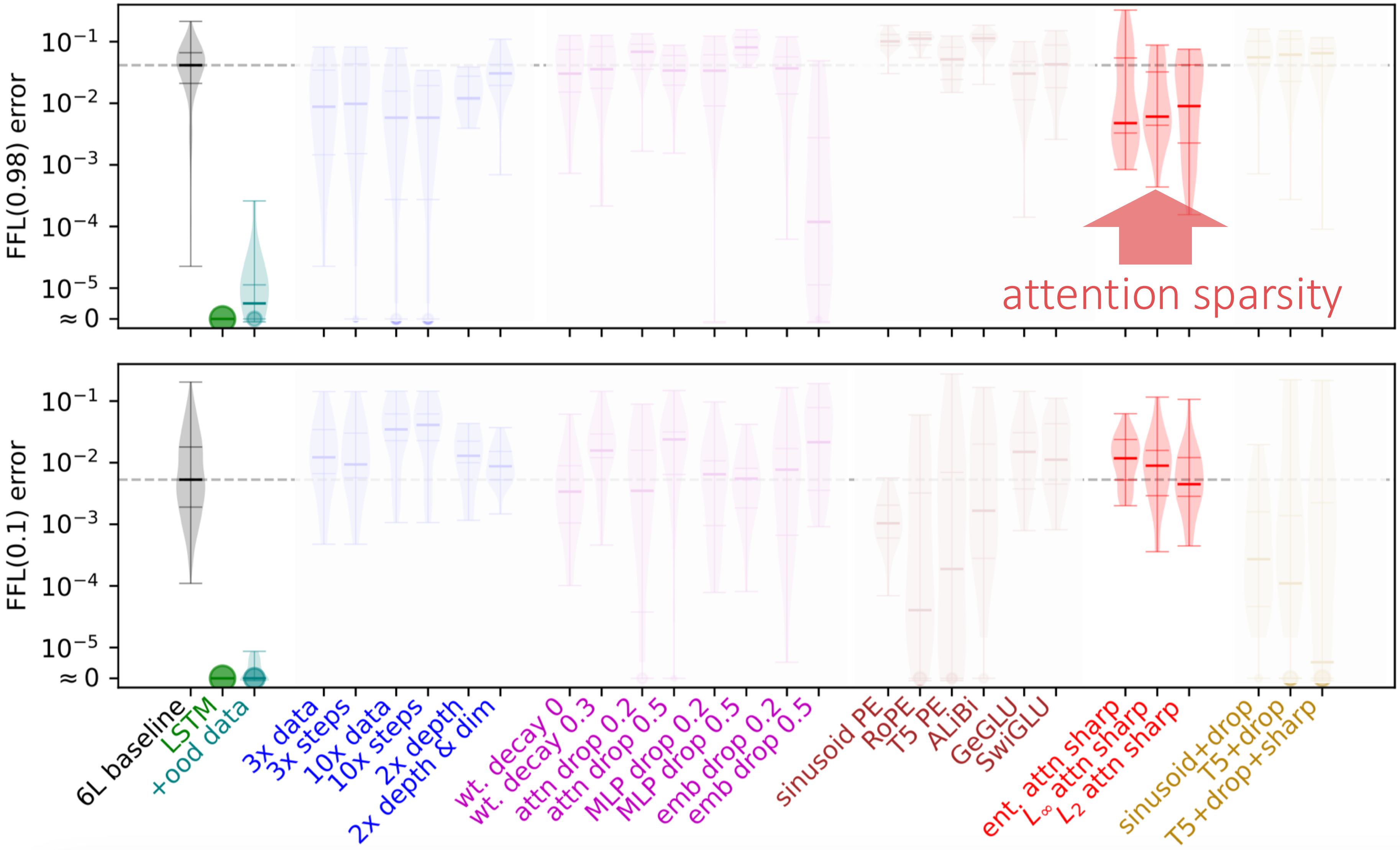
Surprisingly hard to fix: no mitigation helps with *both*



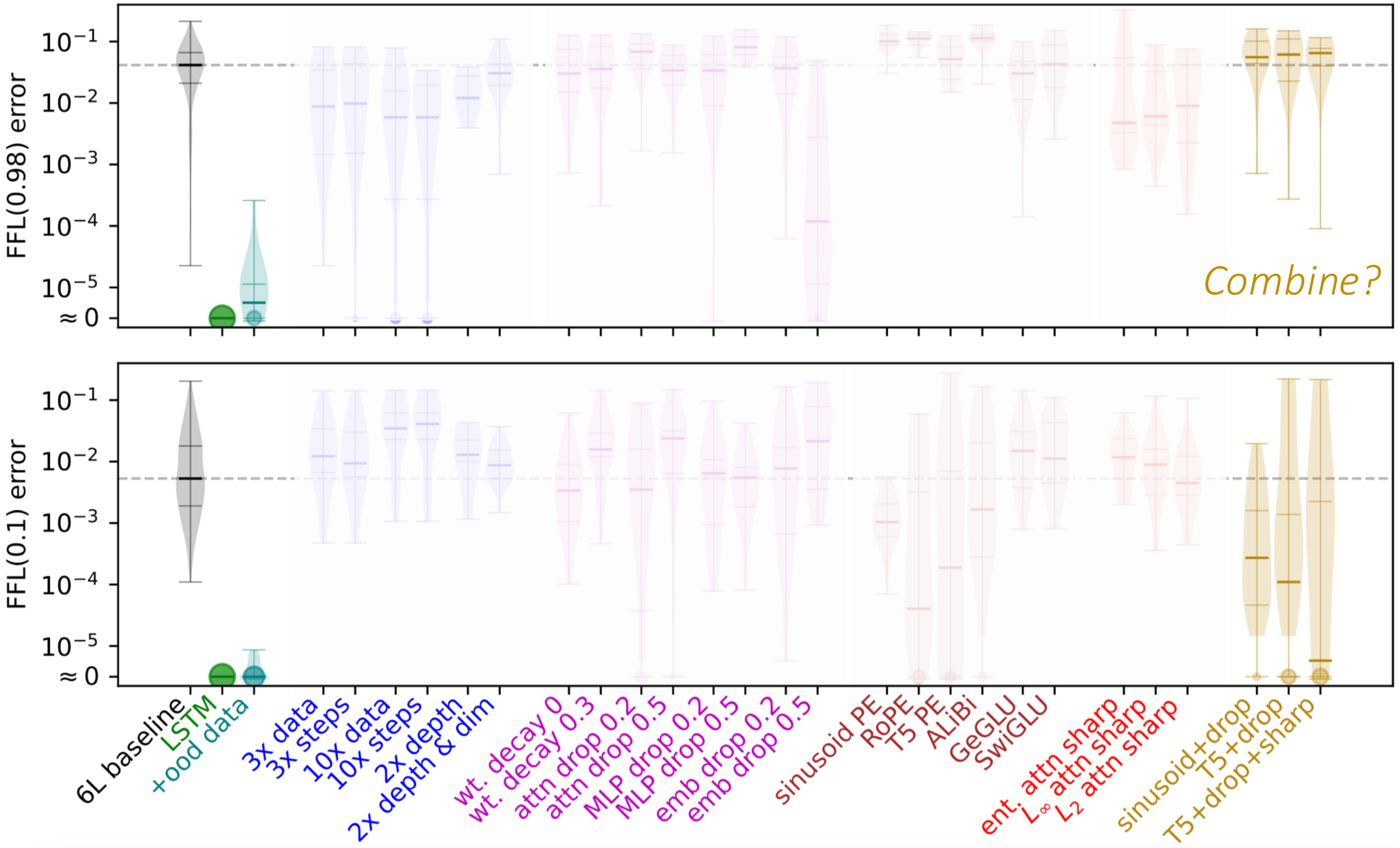
Surprisingly hard to fix: no mitigation helps with *both*



Surprisingly hard to fix: no mitigation helps with *both*



Surprisingly hard to fix: no mitigation helps with *both*



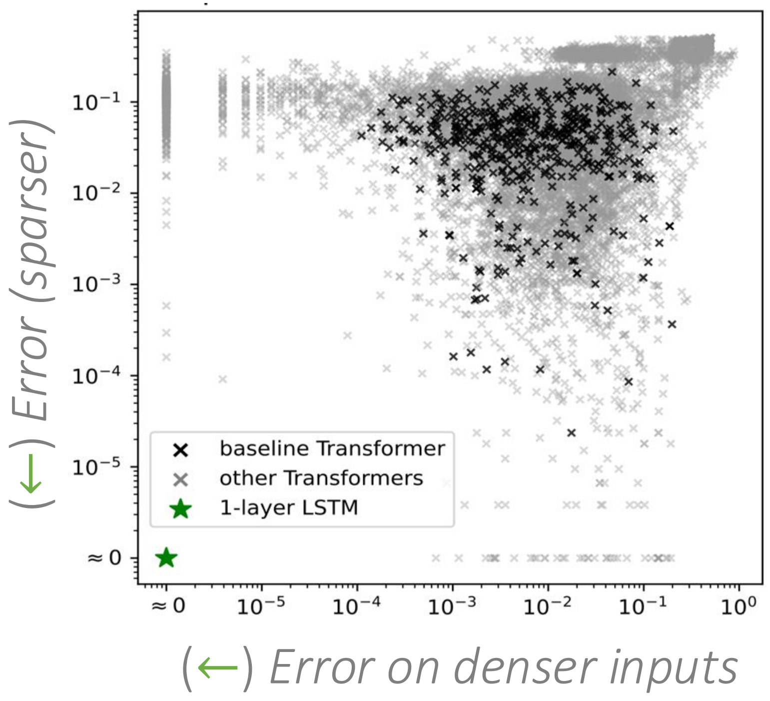
OOD error: attention glitches

Transformer's imperfect hard retrieval.

- Transformers exhibit a long tail of errors, even on an *extremely simple* task.
  - *Goal: learn as well as LSTM?*
- Two inherent limitations of Transformers.
  - Imply various errors; no good mitigation.
- Scaling is no panacea. Data matters.
  - *… recurring theme in the program.*

Flip-Flop Language (FFL)

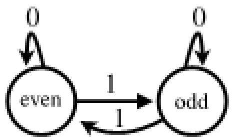
`w 1 i 0 r 1 w 0 i 1 r 1`



Summary

0. Formalizing sequential reasoning.

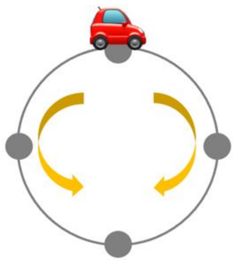
Finite-state automata: $\mathcal{A} = (Q, \Sigma, \delta)$

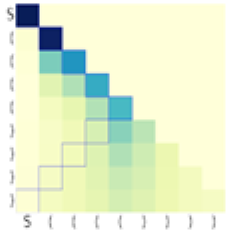


1. **Capabilities** – Shallow solutions to sequential tasks.

$O(\log T)$, $\tilde{O}(|Q|^2)$*-layer "shortcuts" for* T *transitions,*

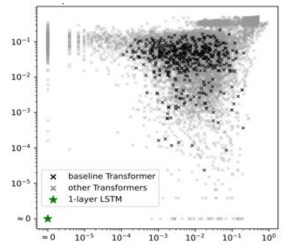
among infinitely many solutions.





2. **Limitations** – Imperfect out-of-distribution performance.

Inherent limitations of Transformers. ***Data*** *is key.*



Proper *abstraction/"sandbox"*
for bridging theory and practice

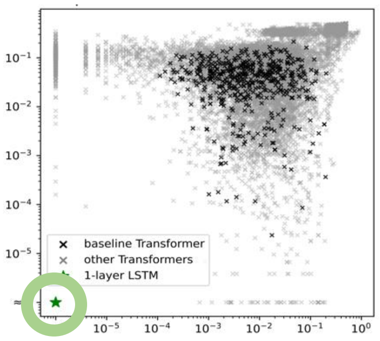
1. Connect to classic theory toolkits
   for understanding modern ML.

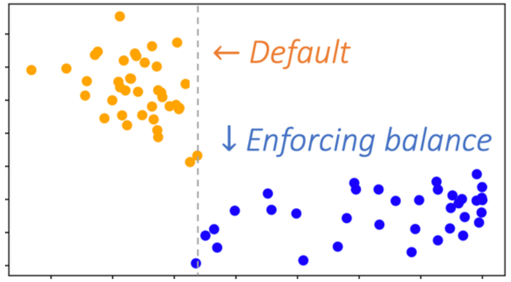
   Representability, various design choices.

   - Automata theory
   - Formal languages
   - Circuit complexity
   - Communication complexity

2. Lightweight experiments for
   (theory-inspired) practical insights.

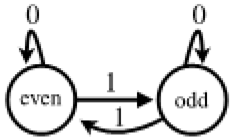
   Diagnoses and mitigations.





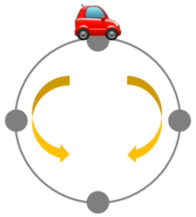
... *and vice versa!* ... e.g. a $O(1)$-layer solution

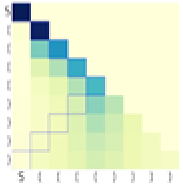
Capabilities & Limitations of Transformers in Sequential Reasoning

0. Formalizing reasoning with $\mathcal{A} = (Q, \Sigma, \delta)$. 

1. **Capabilities** – Shallow solutions to sequential tasks.

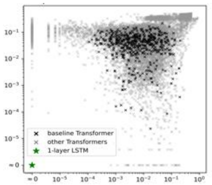
   $O(\log T)$, $\tilde{O}(|Q|^2)$*-layer "shortcuts", among ∞ solutions.*





2. **Limitations** – Imperfect out-of-distribution performance.

   *Inherent limitations of Transformers. **Data** is key.*




Jordan T. Ash


Surbhi Goel


Akshay Krishnamurthy


Yuchen Li


Andrej Risteski


Kaiyue Wen


Cyril Zhang