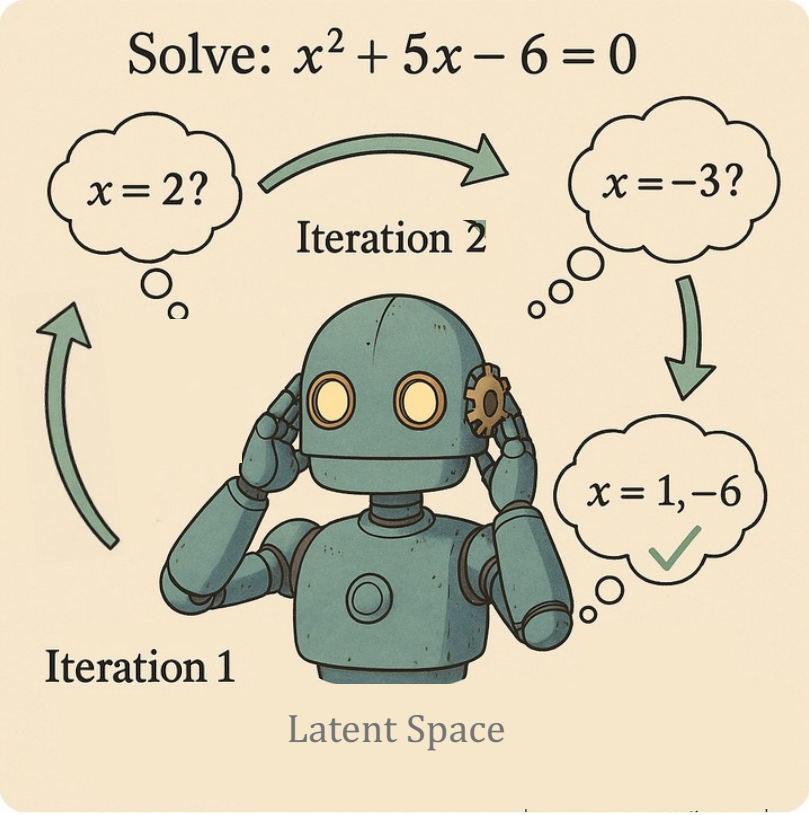
Implicit Language Models are RNNs: Balancing Parallelization and Expressivity

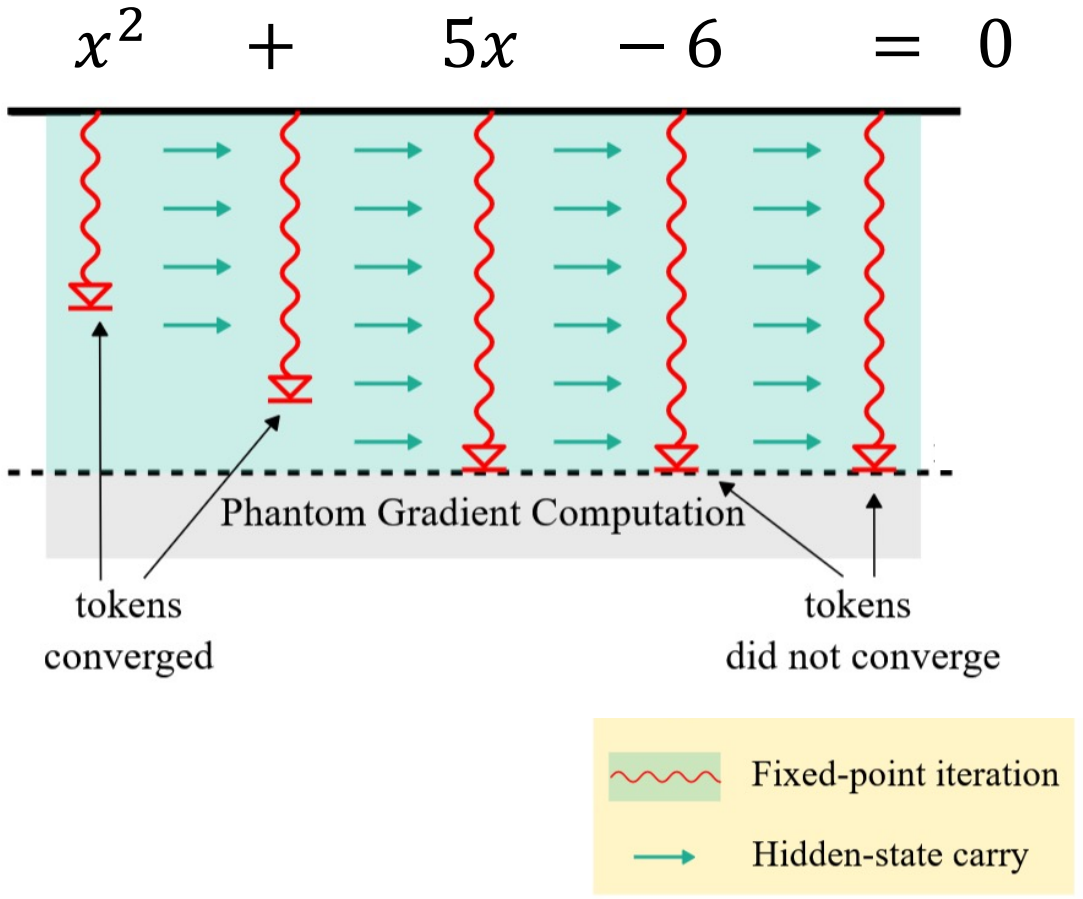
Babak Rahmani*,
M Schöne*, H Kremer,
F Falck, H Ballani, J Gladrow
(*equal contribution)



Microsoft Research

Implicit Language Modelling

- Expressivity
- Parallel Training
- Scalable



Reasoning with LLMs

Puzzles

“swap ball 1 and 3, swap ball 3 and 5, swap ball 4 and 2, ..”

Where does ball 5 end up?

Math

$$\underbrace{(2,4,5,3,1)}_{\sigma_0} \circ \underbrace{(1,3,5,4,2)}_{\sigma_1} = \underbrace{(2,4,5,3,1)}_{\sigma_2}$$

$$\underbrace{(2,4,5,3,1)}_{\sigma_2} \circ \underbrace{(5,4,2,3,1)}_{\sigma_3} = \underbrace{(1,3,4,5,2)}_{\sigma_4}$$

$\text{LLM}(\sigma_{i-1}; \sigma_i) \rightarrow \sigma_{i+1}$

Can LLMs learn the algorithm to calculate the next element?

Code Evaluation

```python
# Assing variables
x = 3
y = 5

# Manipulate variables
x = x + 2
y = y * 2

# Loop
for i in range(3):
    x += i
    y -= 1

print("Final x:", x)
print("Final y:", y)

```

S_5 Permutation composition

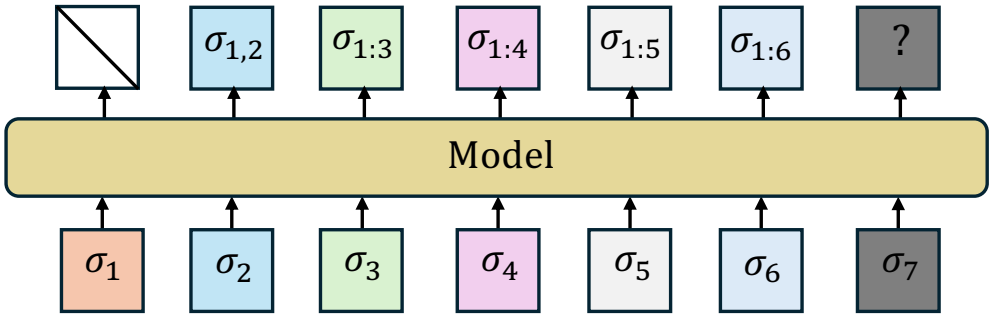
Example

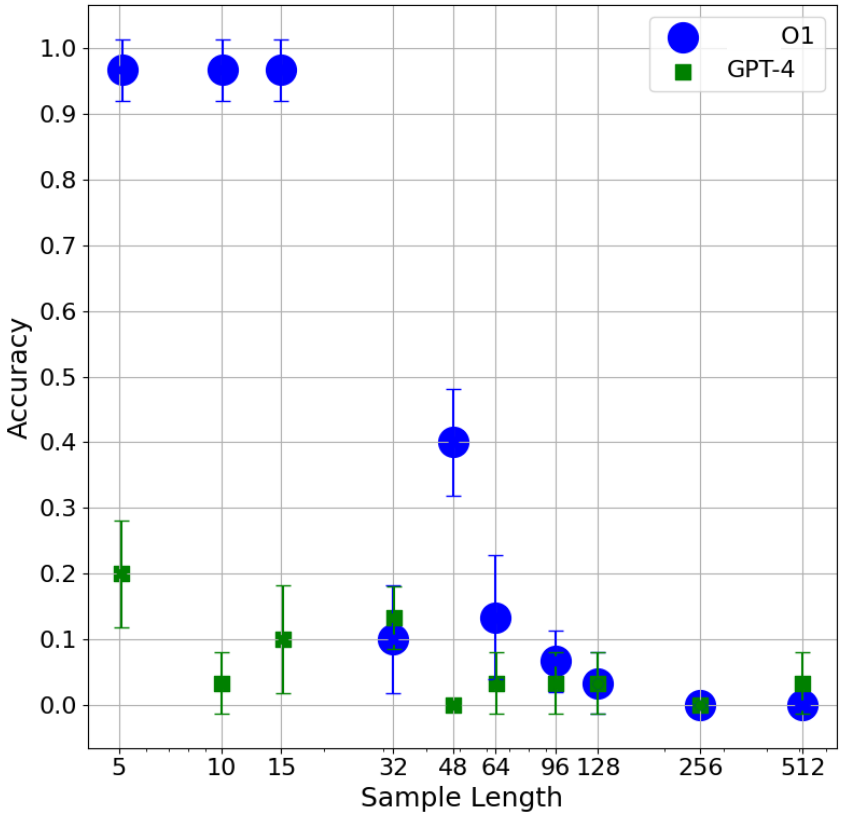
Consider the sequence in S_5:
$\sigma_1 = (5, 4, 1, 3, 2)$, $\sigma_2 = (3, 4, 1, 2, 5)$, and $\sigma_3 = (5, 3, 1, 2, 4)$.
The prefix product is computed as follows:

$$\sigma_{1,2} = \sigma_2 \circ \sigma_1 = (1, 3, 5, 4, 2),$$

$$\sigma_{1,2,3} = \sigma_3 \circ \sigma_{1,2} = (2, 5, 1, 3, 4).$$

Hence, the final permutation is $\sigma_{1,2,3} = (2, 5, 1, 3, 4)$.
Task: Given the sequence: <**SEQ**>, predict the next element in the sequence.





S_5 Permutation composition

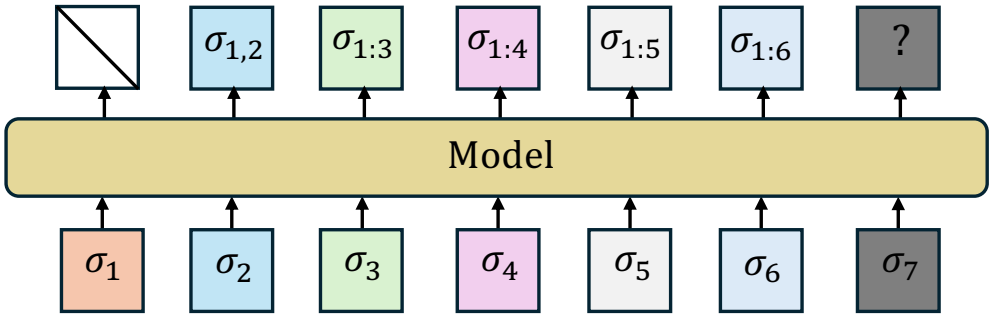
Example

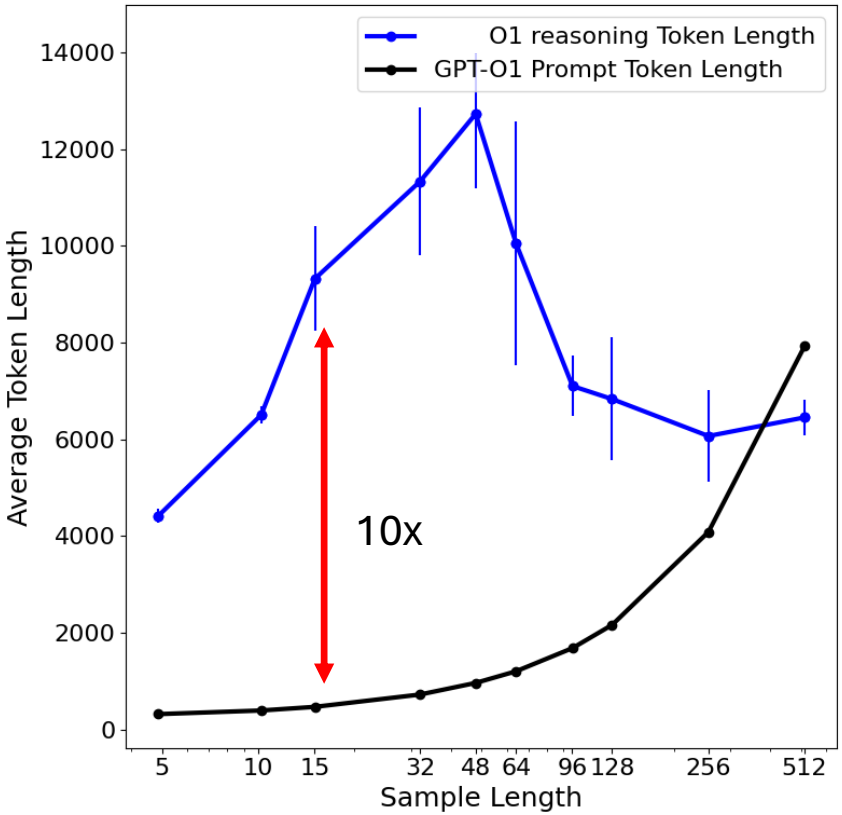
Consider the sequence in S_5:
$\sigma_1 = (5, 4, 1, 3, 2)$, $\sigma_2 = (3, 4, 1, 2, 5)$, and $\sigma_3 = (5, 3, 1, 2, 4)$.
The prefix product is computed as follows:

$$\sigma_{1,2} = \sigma_2 \circ \sigma_1 = (1, 3, 5, 4, 2),$$

$$\sigma_{1,2,3} = \sigma_3 \circ \sigma_{1,2} = (2, 5, 1, 3, 4).$$

Hence, the final permutation is $\sigma_{1,2,3} = (2, 5, 1, 3, 4)$.
Task: Given the sequence: <**SEQ**>, predict the next element in the sequence.



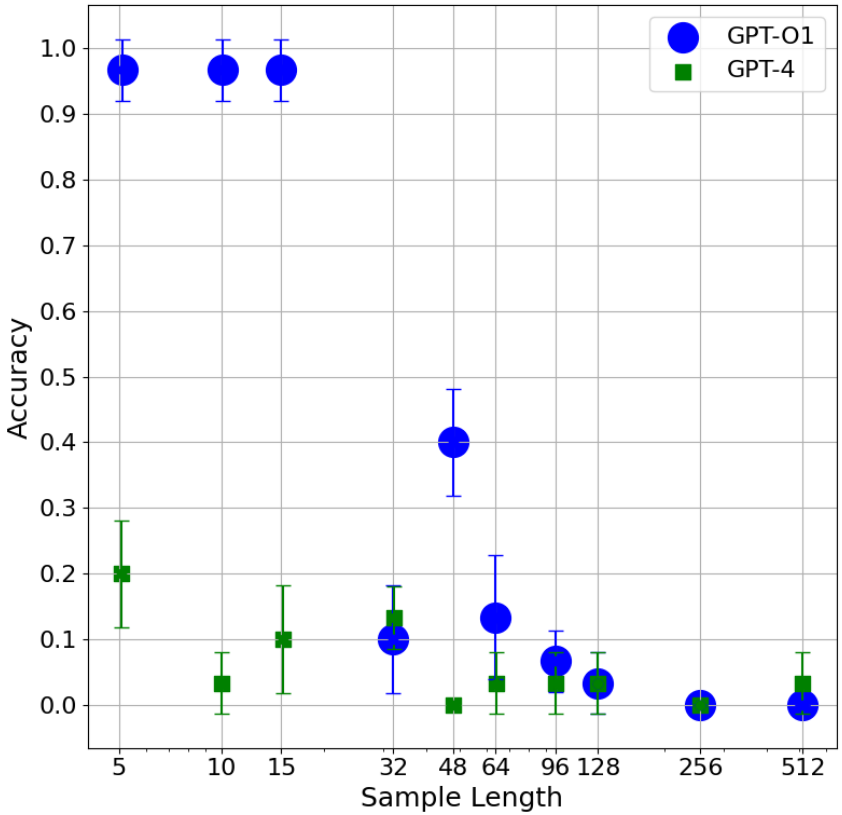


GPT-like models cannot track state

Transformer models are **fixed-depth**

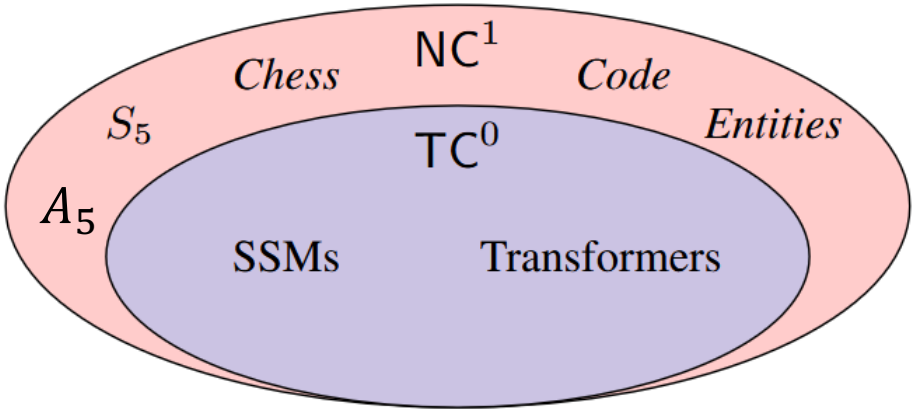
- This **provably limits** what they can express
- Notably, they **cannot** (really) **track state***

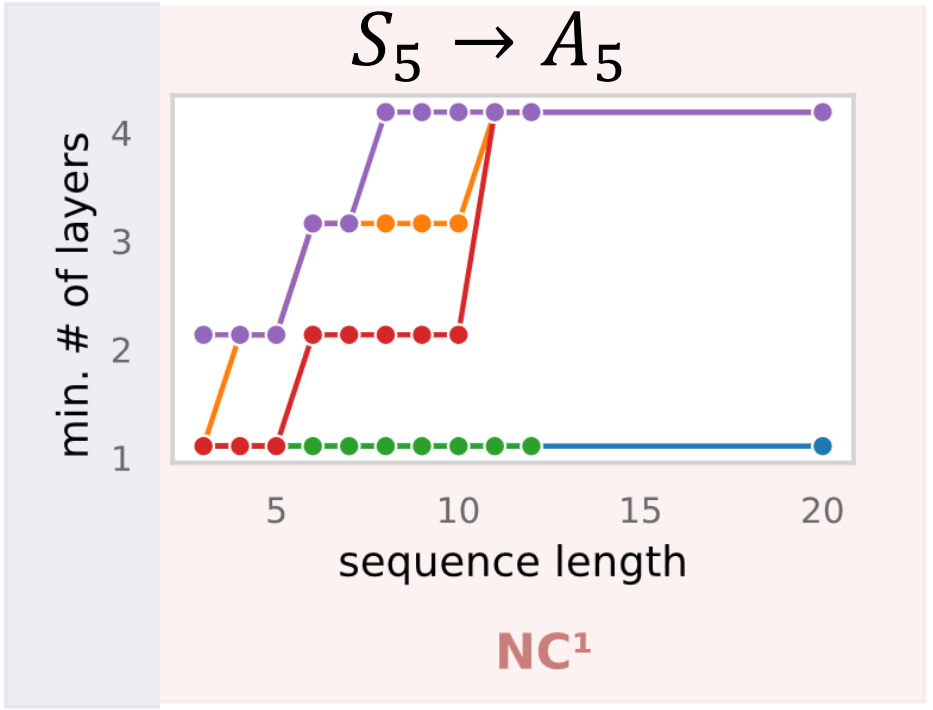
*They often learn *shortcuts*



A Formal Hierarchy of RNN Architectures
Merrill, W., et al 2020
The Illusion of State in State-Space Models
Merrill, W., et.al 2024

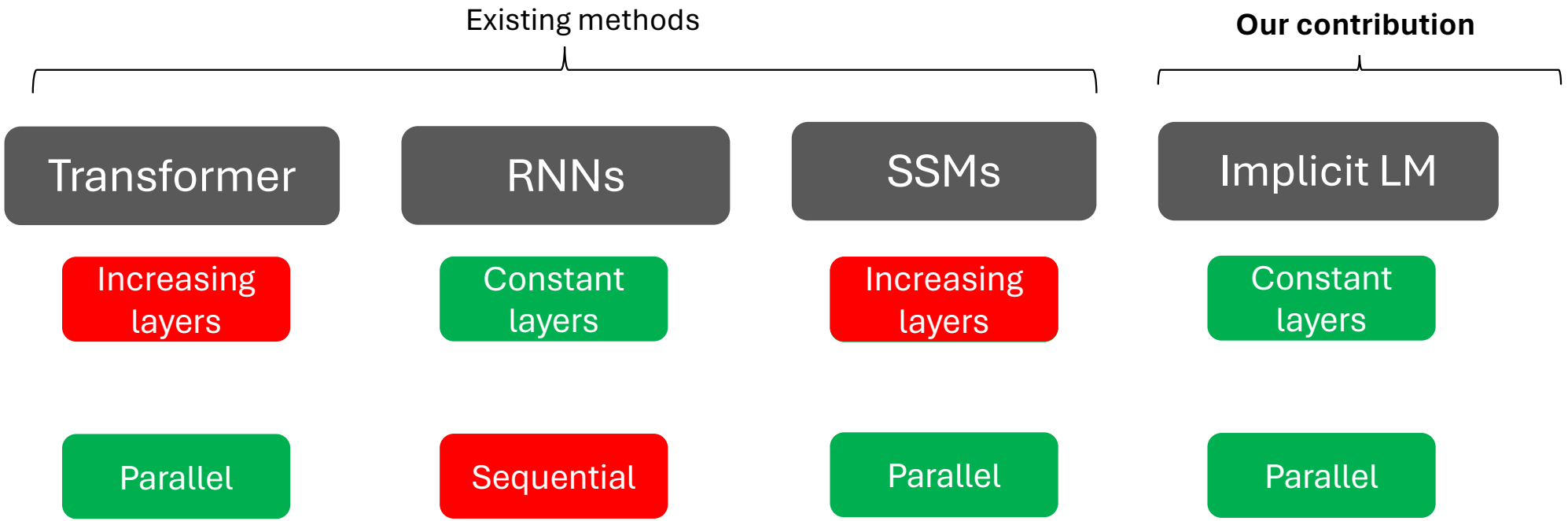
What does it take to solve this task?





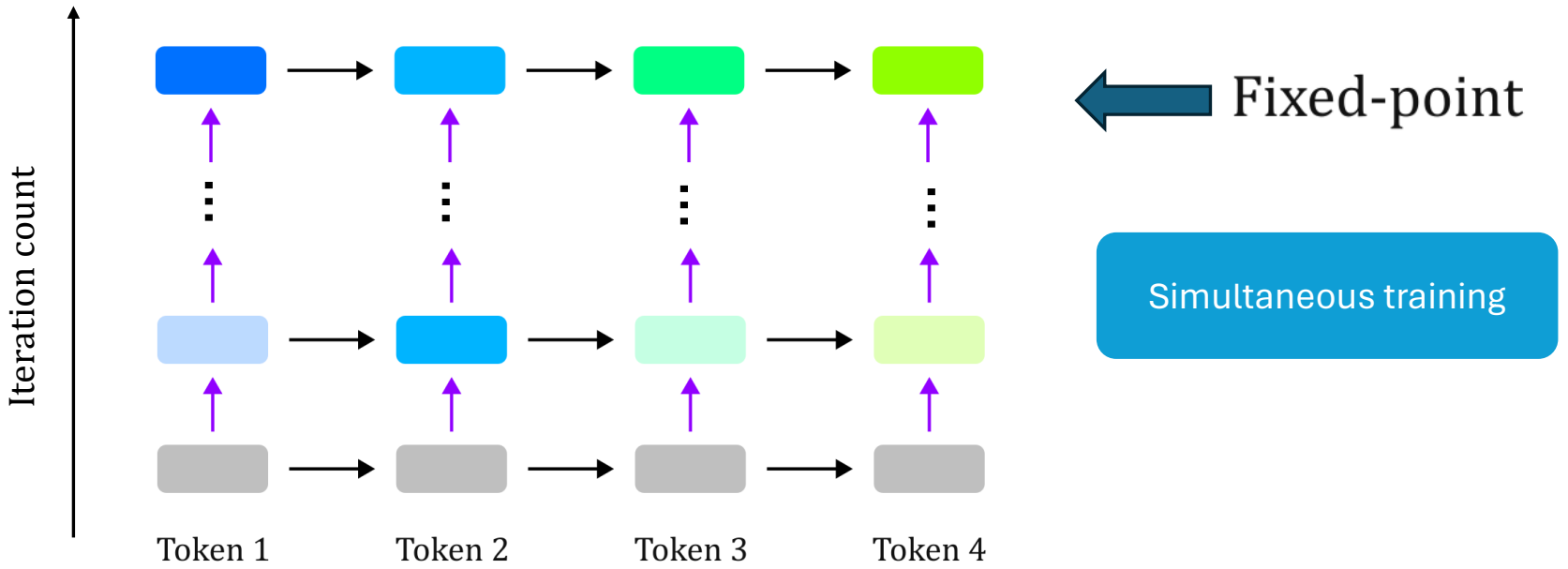
The Illusion of State in State-Space Models
Merrill, W., et.al 2024

What does it take to solve this task?

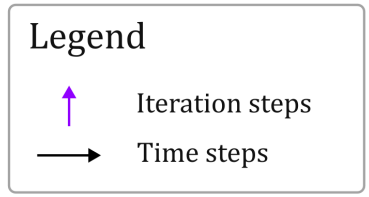


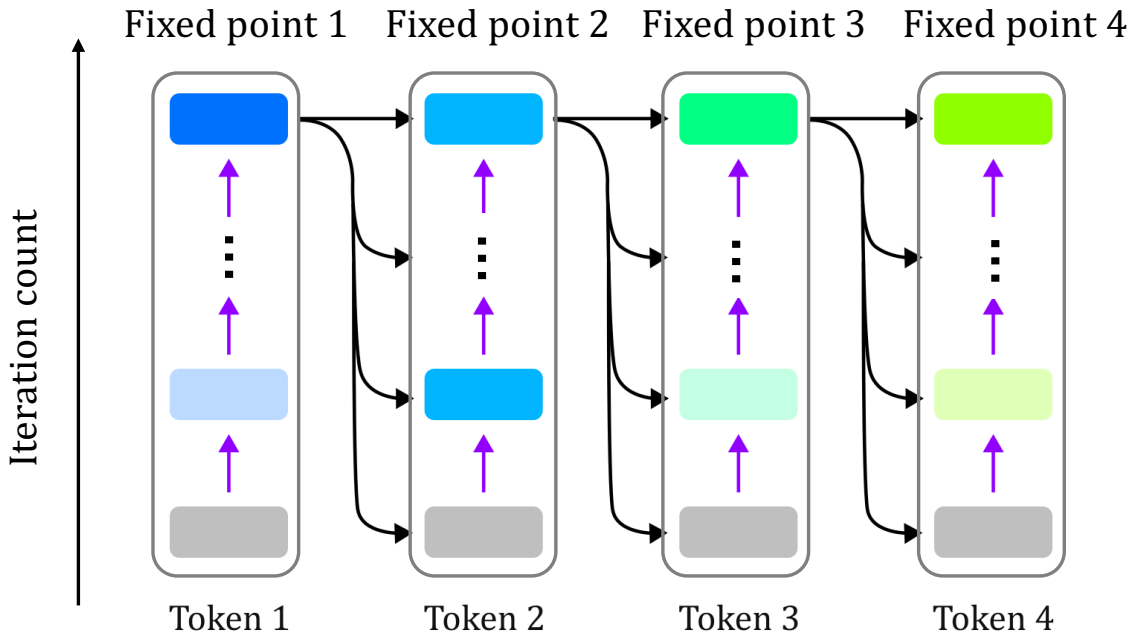
A Formal Hierarchy of RNN Architectures
Merrill, W., et al 2020
The Illusion of State in State-Space Models
Merrill, W., et.al 2024

Implicit State Space Models



Implicit State Space Models

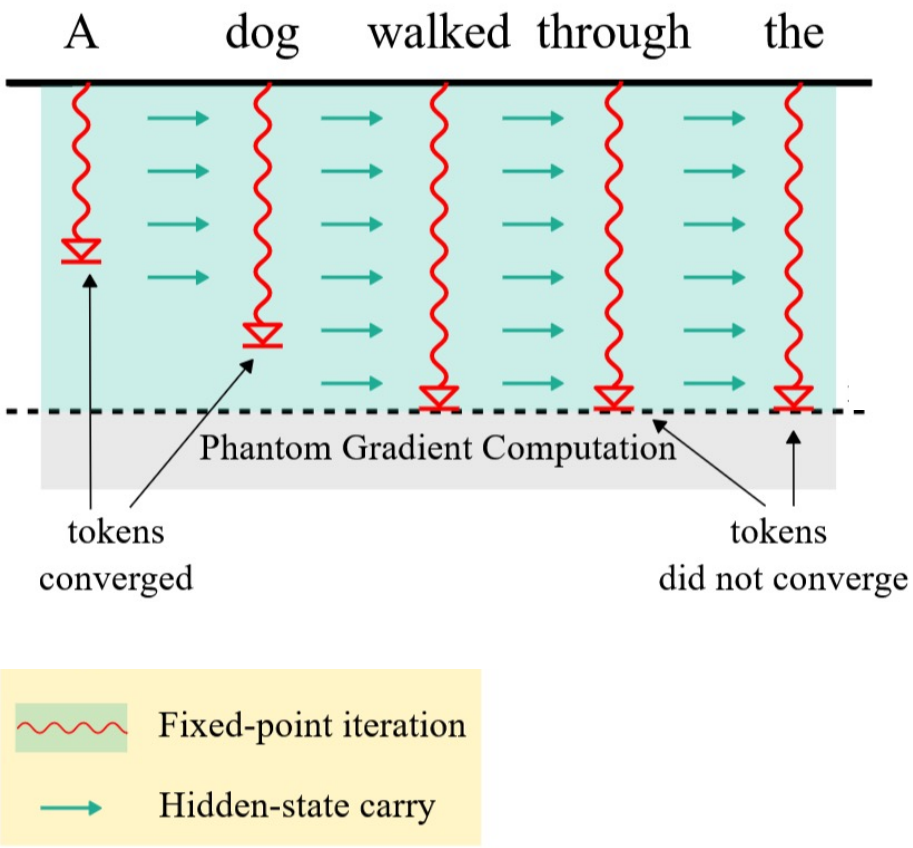


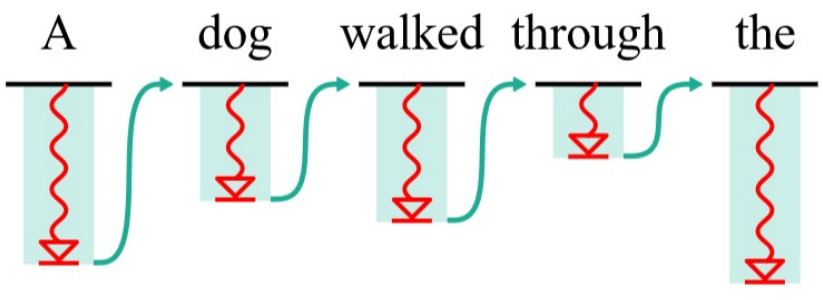


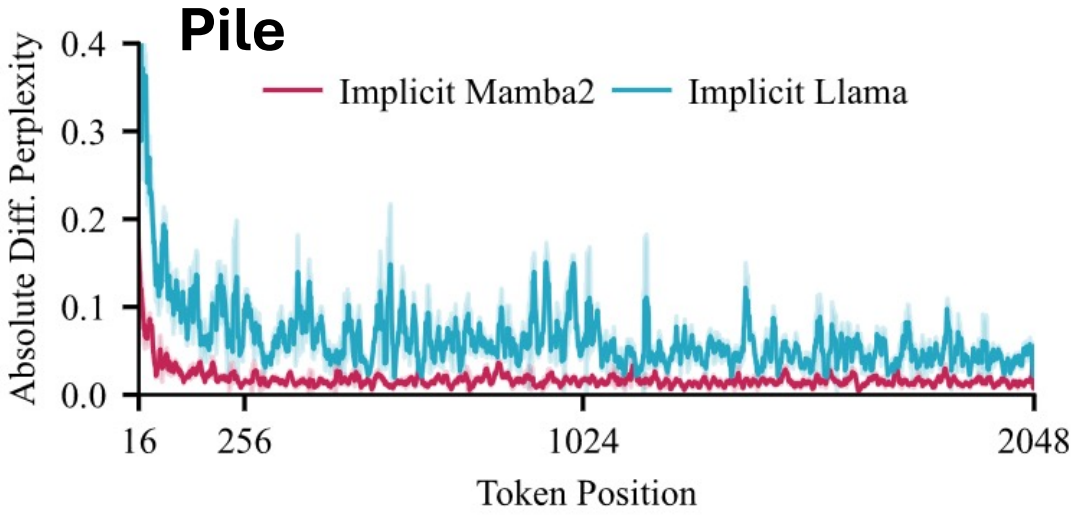
Sequential inference

One/multi token fixed-point inference

Duality between parallel and sequential mode



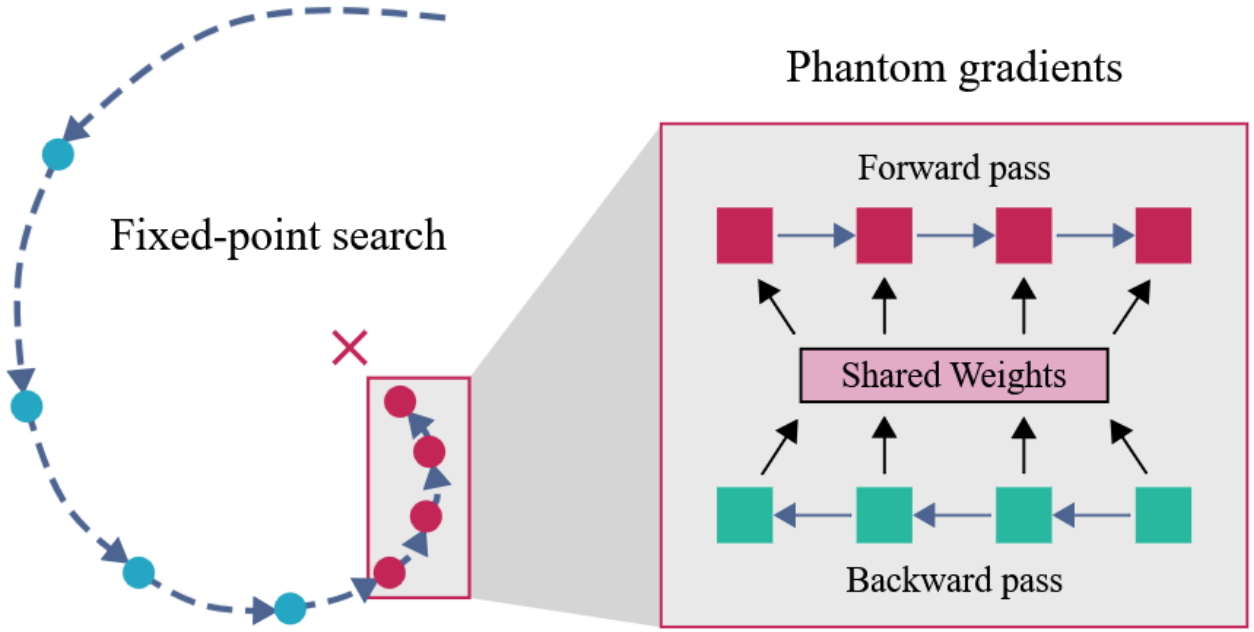




Training Implicit Models

One training step

- 1- Fixed-point search:
  - $z^s = f(z^{s-1}, x)$
- 2- Phantom gradients:
  - $z^{s+1} = (1-\lambda)z^s + \lambda f(z^s, x)$



On Training Implicit Models
Geng z et. al. 2022

Why Implicit LLMs are RNNs?

$$h_t = \Lambda(x_t) h_{t-1} + u(x_t)$$

$$y_t = f_\theta(h_{t-1}, x_t)$$

$$\begin{cases} h_t^{(s)} = \Lambda\left(z_{t-1}^{(s-1)}, x_t\right) h_{t-1}^{(s)} + u\left(z_{t-1}^{(s-1)}, x_t\right) & \text{Eq.1} \\ z_{t-1}^{s} = f_\theta\left(z_{t-1}^{(s-1)}, h_t^{(s)}, x_t\right) & \text{Eq.2} \end{cases}$$

$$\begin{cases} h_t^* = \Lambda(z_{t-1}^*, x_t)\, h_{t-1}^* + u(z_{t-1}^*, x_t) & z_t^* = \lim_{s\to\infty} z_t^s \\ z_{t-1}^* = f_\theta(z_{t-1}^*, h_t^*, x_t) & h_t^* = \lim_{s\to\infty} h_t^s \end{cases}$$

Theorem 1. Consider an implicit SSM defined by equations (1) and (2). Then the transition function $h_{t-1}^{(*)} \to h_t^{(*)}$ is **non-linear** and **non-diagonal**. Consequently, the state-to-state Jacobian is a **non-diagonal** operator.

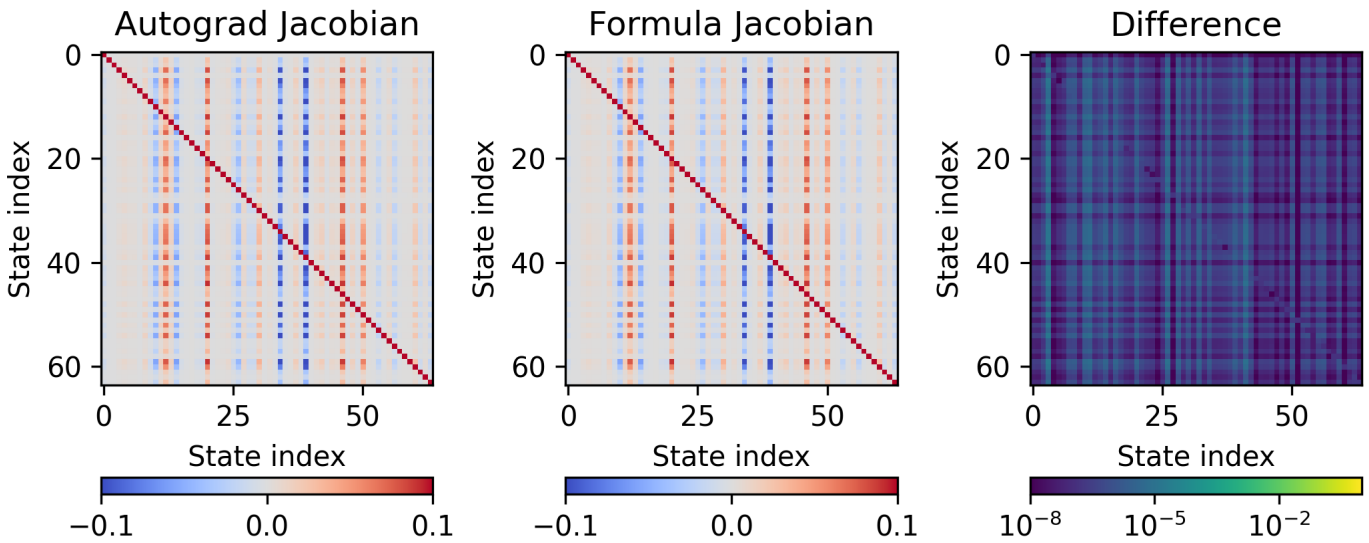
Why Implicit LLMs are RNNs?

Theorem 1. Consider an implicit SSM defined by equations (1) and (2).

Then the transition function $h_{t-1}^{(*)} \to h_t^{(*)}$ is **non-linear** and **non-diagonal**.

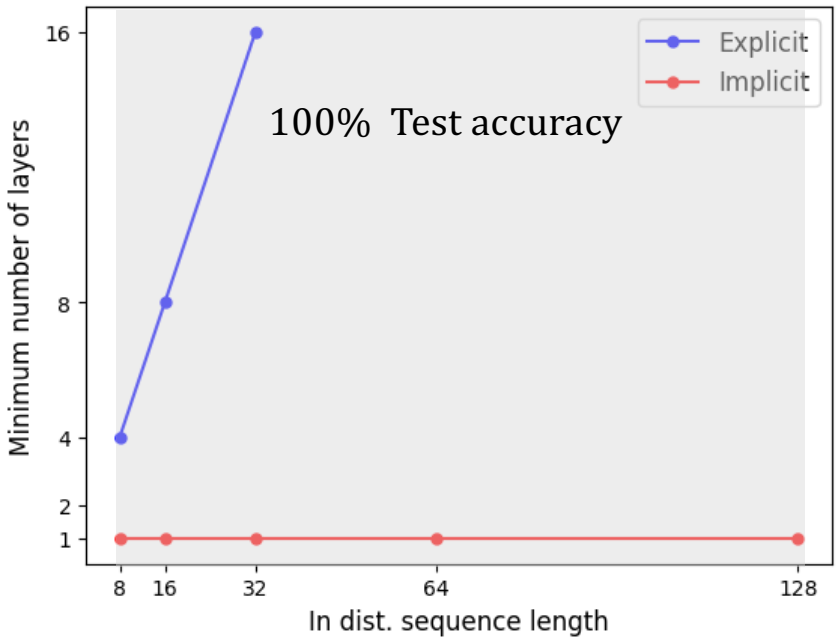
Consequently, the state-to-state Jacobian is a **non-diagonal** operator.

$$\left.\frac{\partial h_t^*}{\partial h_{t-1}^*}\right|_{h_{t-1}^*, x_t, \theta} = \Lambda\left(z_t^*, x_t\right) + \frac{\partial \Lambda}{\partial z_t^*}\frac{\partial \varphi}{\partial h_{t-1}^*}\operatorname{diag}\left(h_{t-1}^*\right) + \frac{\partial u}{\partial z_t^*}\frac{\partial \varphi}{\partial h_{t-1}^*}.$$

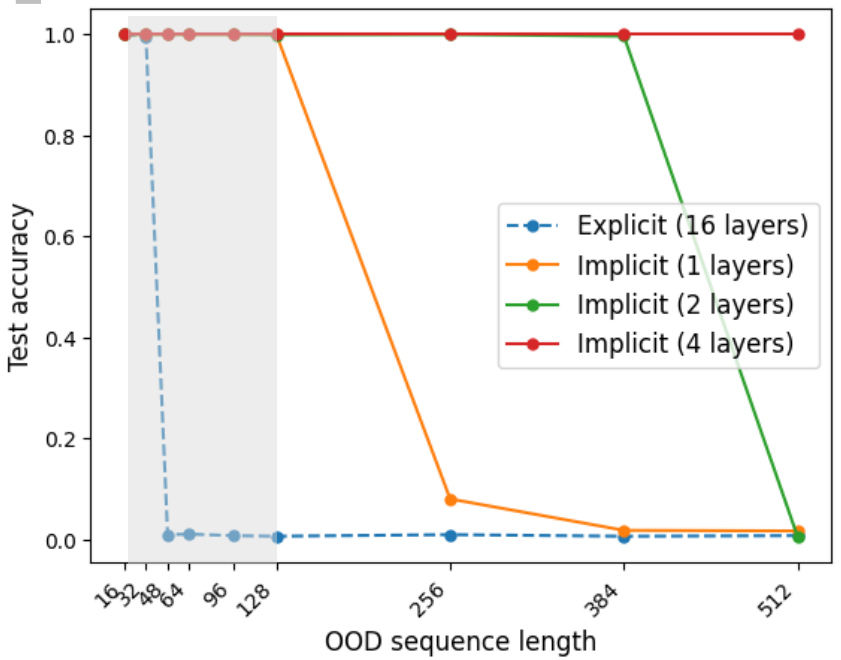


Lifting the “Illusion of state”: Word-problem

better S_5 (in distribution)



better S_5 (out of distribution)



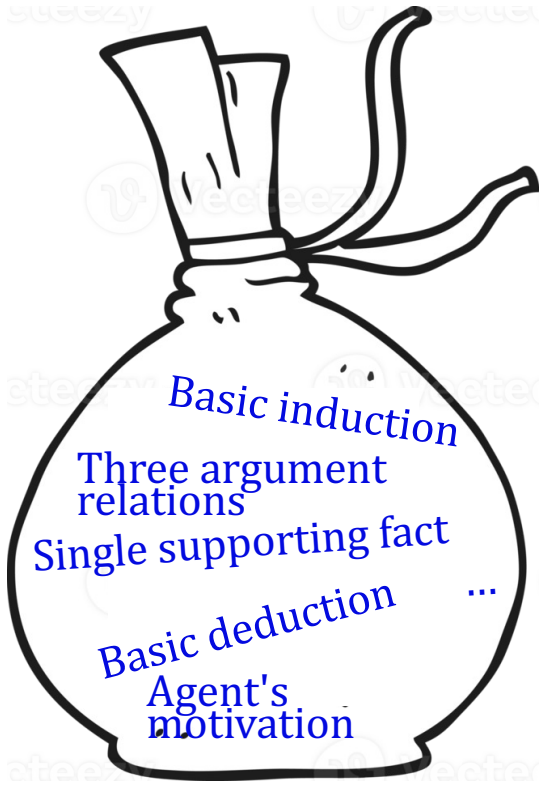
Catbabi dataset: a bag of reasoning tasks

Basic Deduction

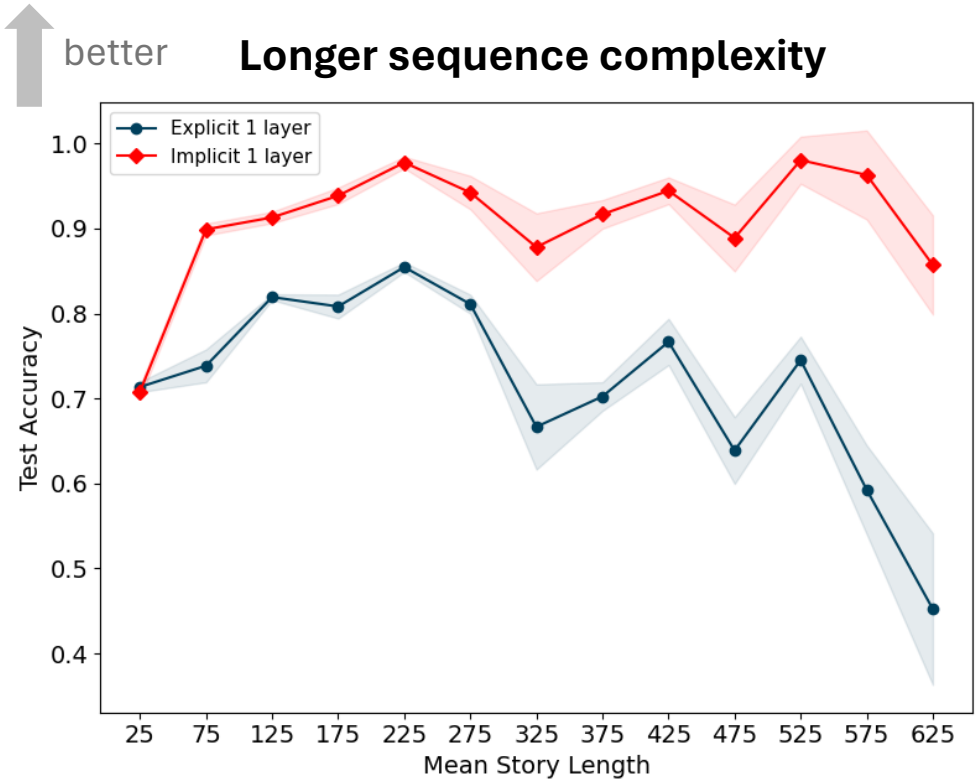
mice are afraid of cats. cats are afraid of wolves. emily is a cat. wolves are afraid of cats. jessica is a mouse. gertrude is a wolf. sheep are afraid of mice. winona is a wolf. what is winona afraid of ? **cat**

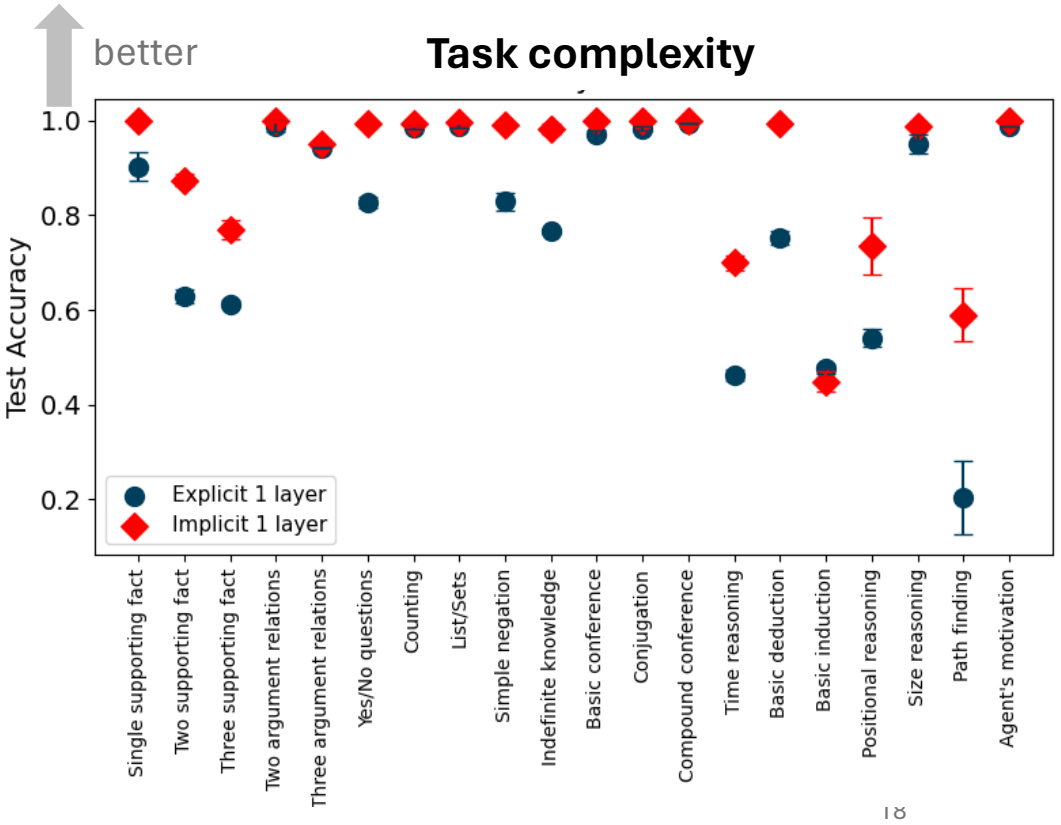
Path finding

the bathroom is south of the garden. the kitchen is north of the bedroom. the hallway is north of the garden. the bedroom is west of the garden. the office is west of the hallway. how do you go from the hallway to the bedroom? **s,w**



Implicit SSMs outperform explicit models in Catbabi reasoning

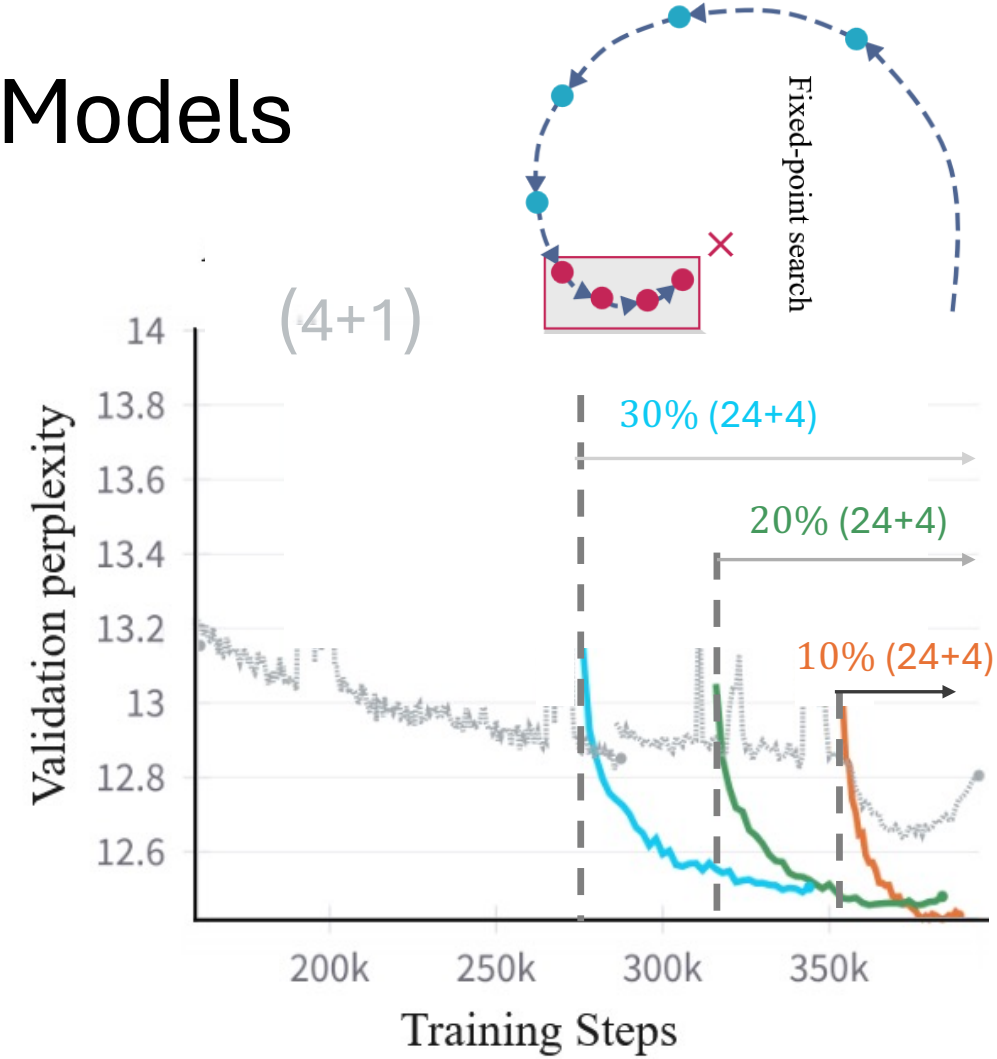




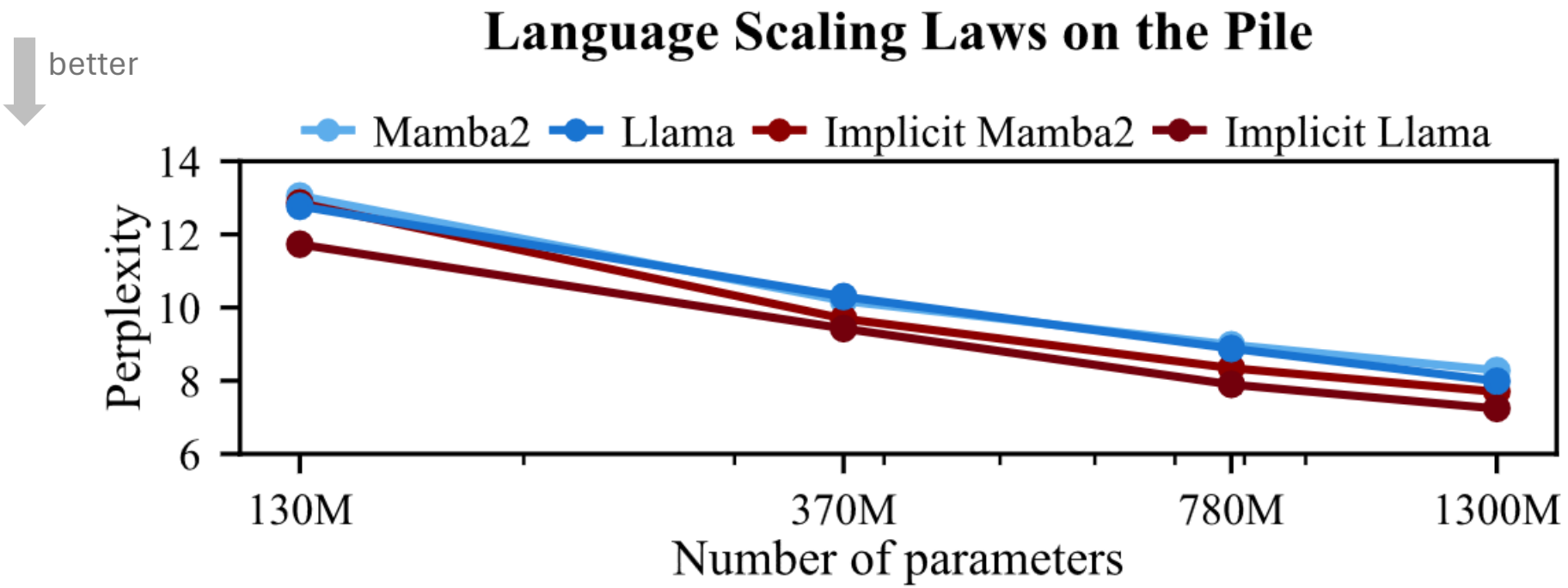
Implicit Large Language Models

Implicit Llama and Mamba2

- Size: **1.3B, 760M, 350M, 130M**
- Dataset: **Pile** (207B tokens)
- Curriculum Training



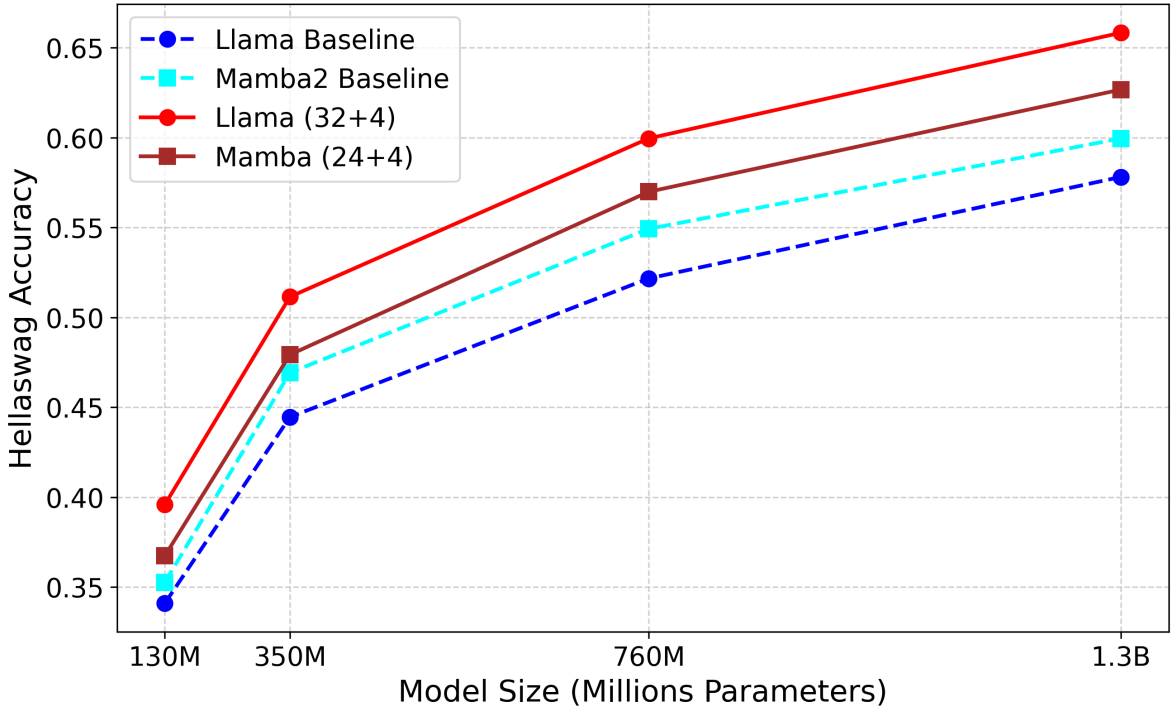
Implicit Large Language Models are Parameter Optimal



DEQ LLMs outperform baselines in HellaSwag

A woman is outside with a bucket and a dog. The dog is running around trying to avoid a bath. She...

A. rinses the bucket off with soap and blow dry the dog's head.
B. uses a hose to keep it from getting soapy.
C. gets the dog wet, then it runs away again.
D. gets into a bath tub with the dog.



Downstream task performances

Llama

	Model	Dataset/Tokens (B)	D-Pile ppl↓	LAMBADA ppl↓	LAMBADA acc↑	HellaSwag acc↑	PIQA acc↑	Arc-E acc↑	Arc-C acc↑	WinoGrande acc↑	OpenbookQA acc↑	Average acc↑
760M	Llama	SlimPajama/300	-	9.90	0.5141	0.5216	0.7095	0.5648	0.2875	0.5667	-	0.5274
	Llama†	D-Pile/207	8.88	5.77	0.6375	0.5448	0.7171	0.5905	0.2816	**0.6054**	0.338	0.5307
	Llama (4+1)-ours	D-Pile/207	8.27	5.15	0.6524	0.5853	0.7312	0.6052	**0.3097**	0.5967	**0.356**	0.5481
	Llama (32+4)-ours	D-Pile/207	**7.90**	**4.82**	**0.6703**	**0.5995**	**0.7416**	**0.6187**	0.3012	0.5991	0.344	**0.5535**
1.3B	Llama	SlimPajama/300	-	7.23	0.5744	0.5781	0.7312	0.6279	0.3174	0.5904	-	0.5699
	Llama†	D-Pile/207	7.99	4.95	0.6569	0.5936	0.7432	0.6385	0.3217	0.6062	0.352	0.5589
	Llama (4+1)-ours	D-Pile/207	7.66	4.40	0.6852	0.6397	0.7448	0.6338	0.3396	**0.6575**	0.360	0.5801
	Llama (32+4)-ours	D-Pile/207	**7.24**	**4.24**	**0.6901**	**0.6583**	**0.7465**	**0.6654**	**0.3601**	0.6401	**0.364**	**0.5892**

Downstream task performances

Llama

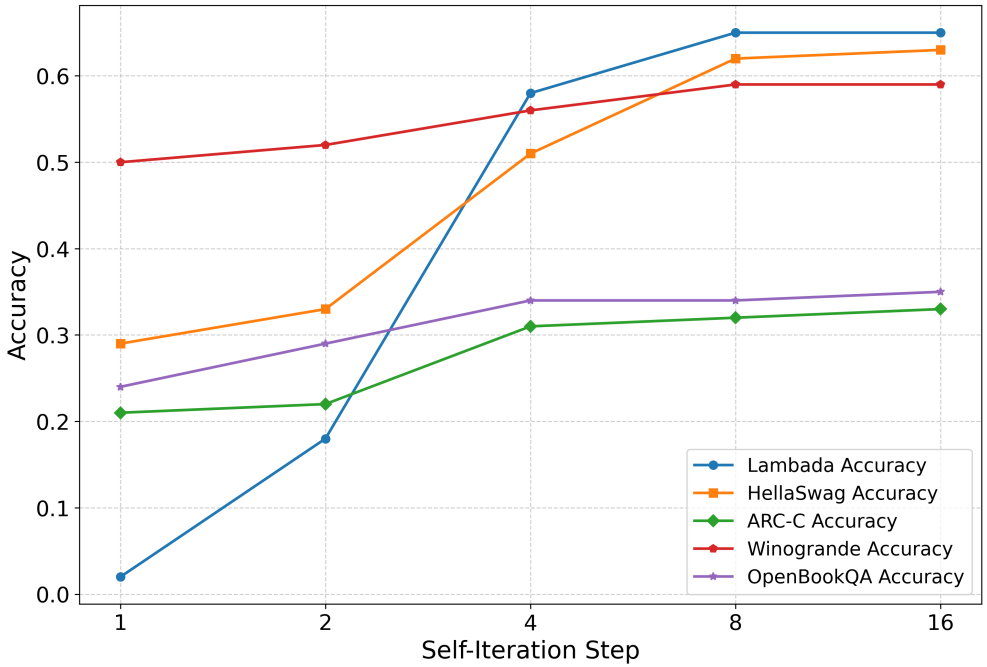
	Model	Dataset/Tokens (B)	D-Pile ppl↓	LAMBADA ppl↓	LAMBADA acc↑	HellaSwag acc↑	PIQA acc↑	Arc-E acc↑	Arc-C acc↑	WinoGrande acc↑	OpenbookQA acc↑	Average acc↑
760M	Llama	SlimPajama/300	-	9.90	0.5141	0.5216	0.7095	0.5648	0.2875	0.5667	-	0.5274
	Llama†	D-Pile/207	8.88	5.77	0.6375	0.5448	0.7171	0.5905	0.2816	**0.6054**	0.338	0.5307
	Llama (4+1)-ours	D-Pile/207	8.27	5.15	0.6524	0.5853	0.7312	0.6052	**0.3097**	0.5967	**0.356**	0.5481
	Llama (32+4)-ours	D-Pile/207	**7.90**	**4.82**	**0.6703**	**0.5995**	**0.7416**	**0.6187**	0.3012	0.5991	0.344	**0.5535**
1.3B	Llama	SlimPajama/300	-	7.23	0.5744	0.5781	0.7312	0.6279	0.3174	0.5904	-	0.5699
	Llama†	D-Pile/207	7.99	4.95	0.6569	0.5936	0.7432	0.6385	0.3217	0.6062	0.352	0.5589
	Llama (4+1)-ours	D-Pile/207	7.66	4.40	0.6852	0.6397	0.7448	0.6338	0.3396	**0.6575**	0.360	0.5801
	Llama (32+4)-ours	D-Pile/207	**7.24**	**4.24**	**0.6901**	**0.6583**	**0.7465**	**0.6654**	**0.3601**	0.6401	**0.364**	**0.5892**

Mamba2

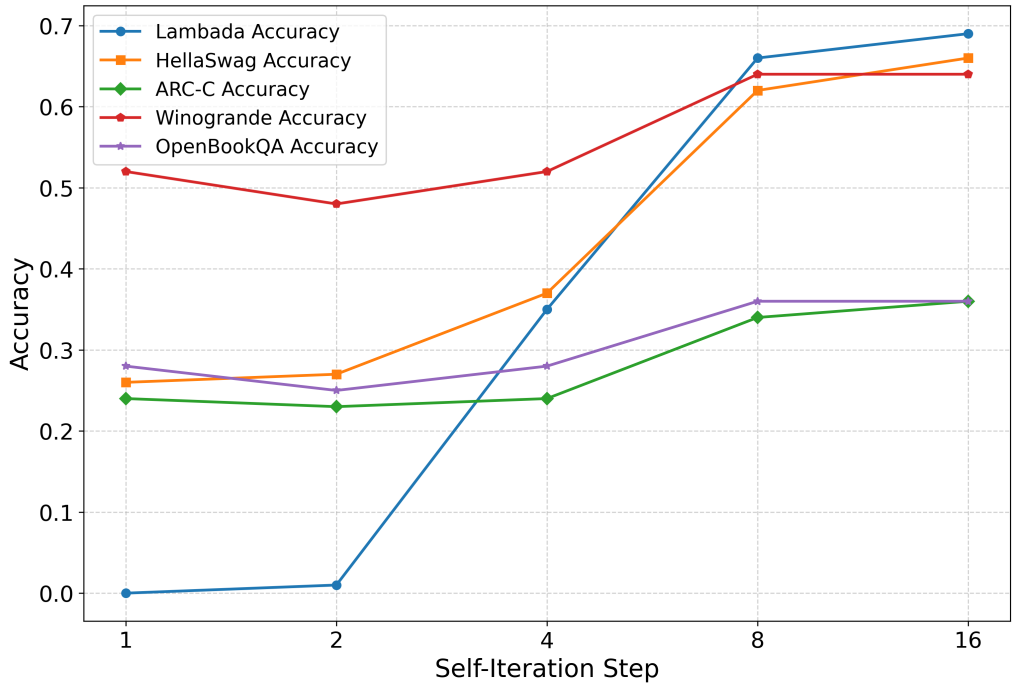
	Model	Dataset/Tokens (B)	D-Pile ppl↓	LAMBADA ppl↓	LAMBADA acc↑	HellaSwag acc↑	PIQA acc↑	Arc-E acc↑	Arc-C acc↑	WinoGrande acc↑	OpenbookQA acc↑	Average acc↑
760M	Mamba2	Pile/300	9.23	**5.86**	0.6167	0.5492	0.7198	**0.6103**	0.2850	**0.6030**	**0.362**	0.5351
	Mamba2*	D-Pile/207	8.98	6.24	0.6125	0.5418	0.7231	0.6044	0.2858	0.5777	0.338	0.5262
	Mamba2(4+1)-ours	D-Pile/207	8.60	6.15	0.6117	0.5569	0.7296	0.6077	**0.3140**	0.5509	0.336	0.5295
	Mamba2(24+4)-ours	D-Pile/207	**8.35**	5.90	**0.6191**	**0.5698**	**0.7334**	0.6090	0.3131	0.5730	0.338	**0.5365**
1.3B	Mamba2	Pile/300	8.40	5.02	**0.6559**	0.5995	0.7378	0.6418	0.3319	**0.6117**	**0.378**	**0.5652**
	Mamba2*	D-Pile/207	8.28	5.12	0.6456	0.5939	0.7416	0.6145	0.3123	0.6117	0.352	0.5531
	Mamba2(4+1)-ours	D-Pile/207	7.97	5.21	0.6383	0.6136	**0.7437**	**0.6343**	0.3302	0.5746	0.354	0.5555
	Mamba2(24+4)-ours	D-Pile/207	**7.70**	**4.99**	0.6489	**0.6267**	0.7416	0.6423	**0.3336**	0.5888	0.352	0.5620

Test-time compute performance

Implicit Mamba2 1.3B



Implicit Llama 1.3B



Summary & Outlook

What we did:

1. Theoretically showed that **Implicit SSMs are RNNs**

2. **Lifting the illusion** of state space models

3. **Scalable Implicit training** up to 1.3B

4. **Improved state tracking** and broader down stream task in llama and Mamba2