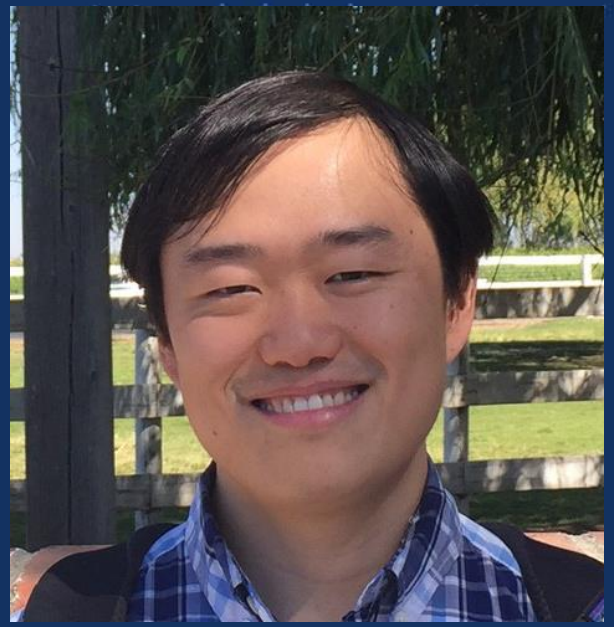
Reason by Search or by Representation? A Path Towards Unifying Neural and Symbolic Decision Making

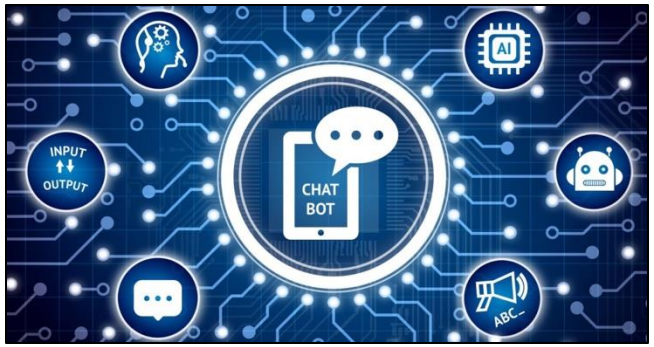
Yuandong Tian
Research Scientist Director

Meta AI



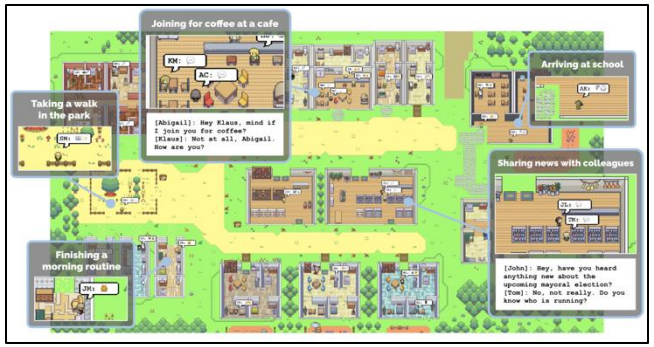
facebook Artificial Intelligence

Large Language Models (LLMs)

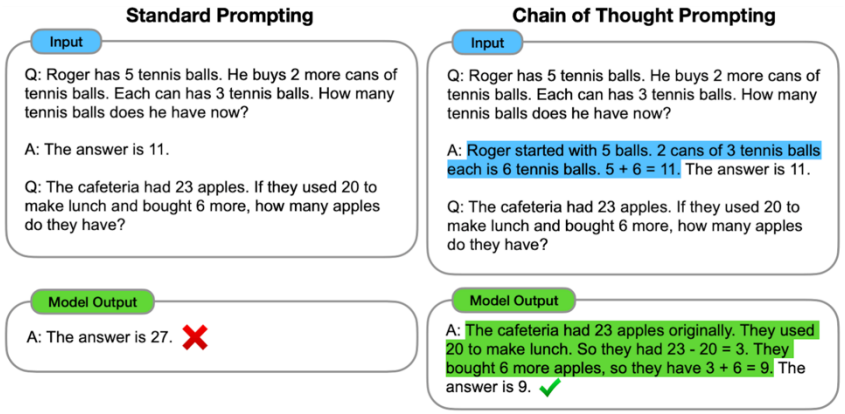


Conversational AI

Content Generation



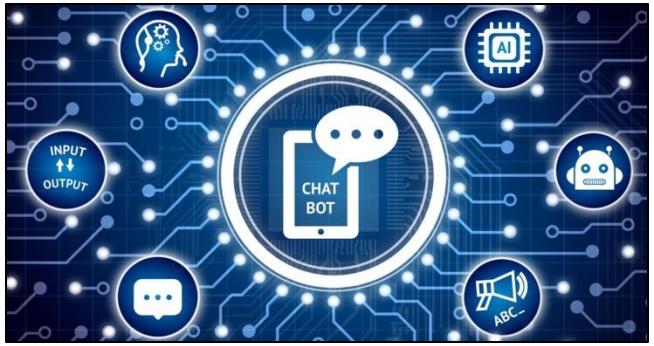
AI Agents



Reasoning

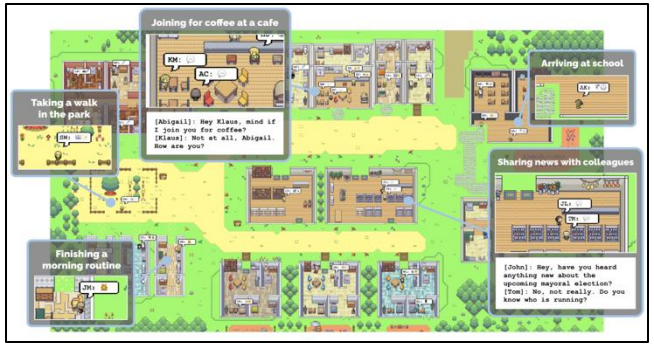
Planning

Large Language Models (LLMs)

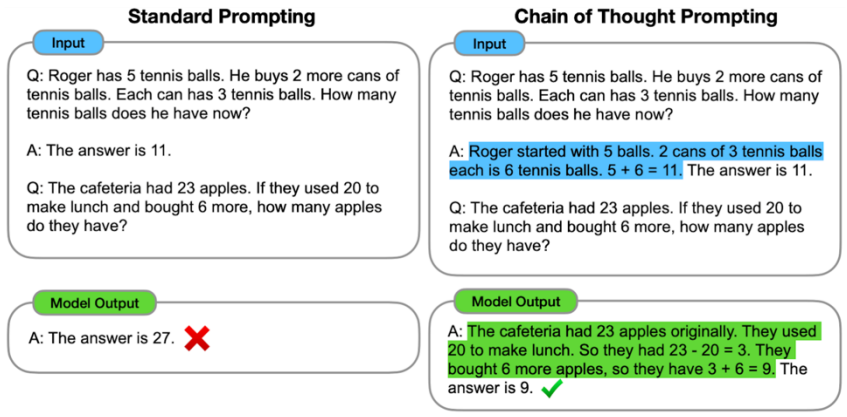


Conversational AI

Content Generation



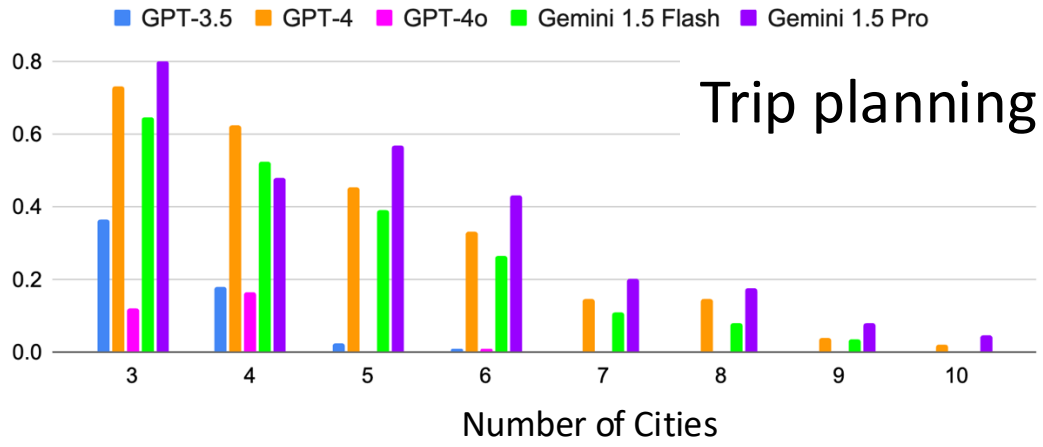
AI Agents

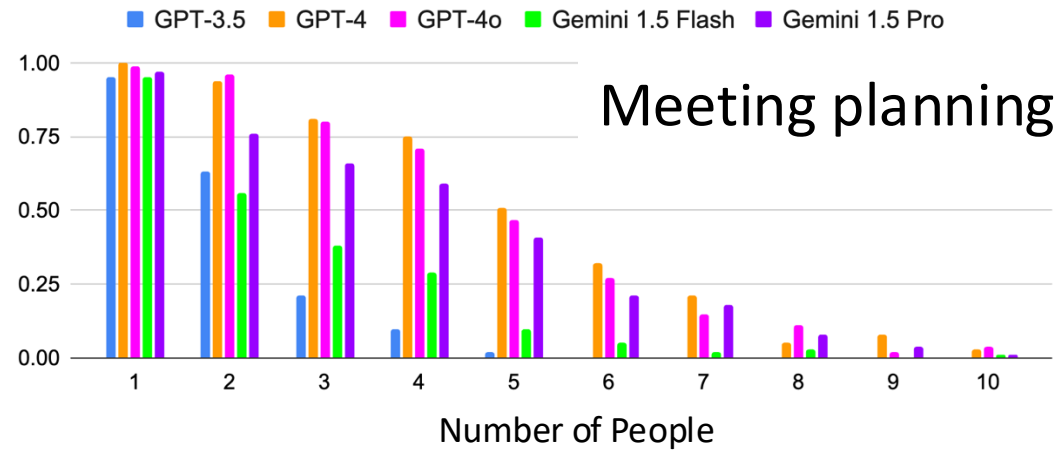


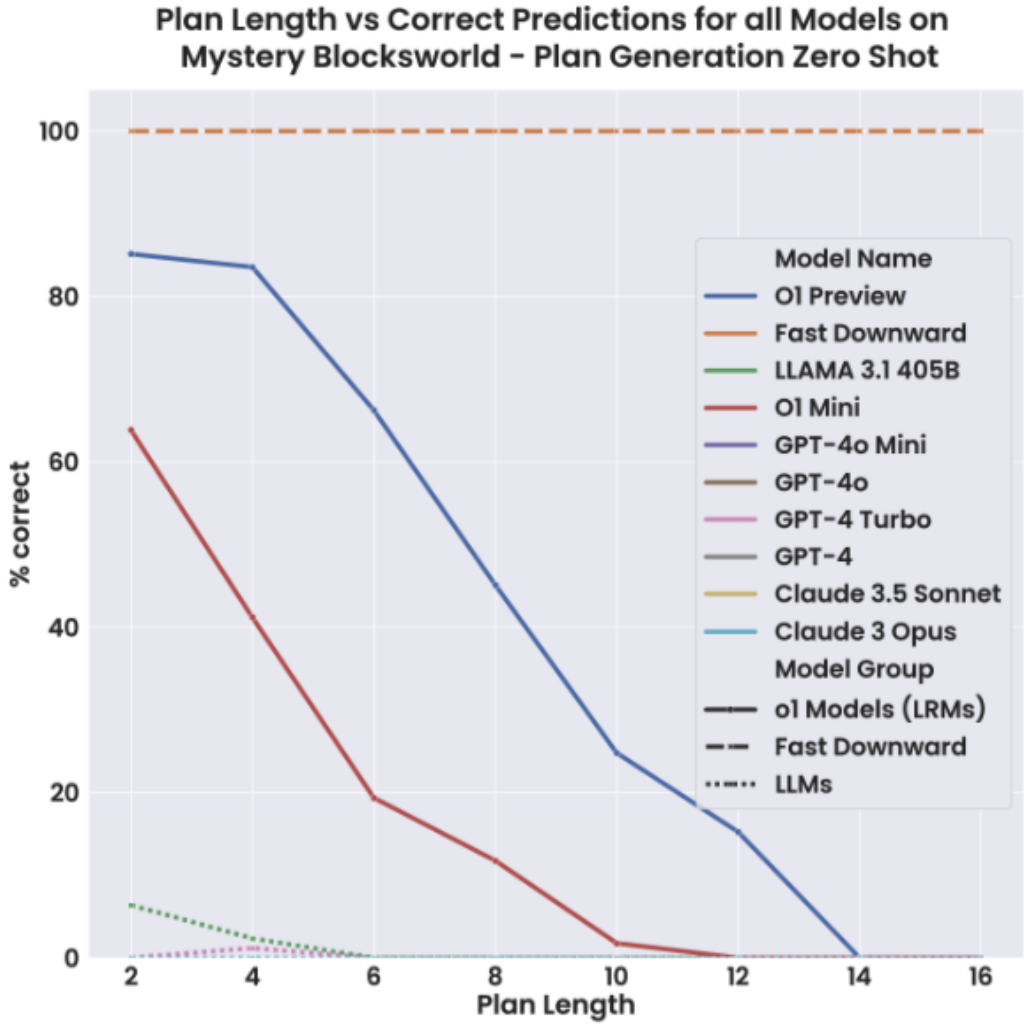
Reasoning

Planning

LLM reasoning & planning is still a hard problem

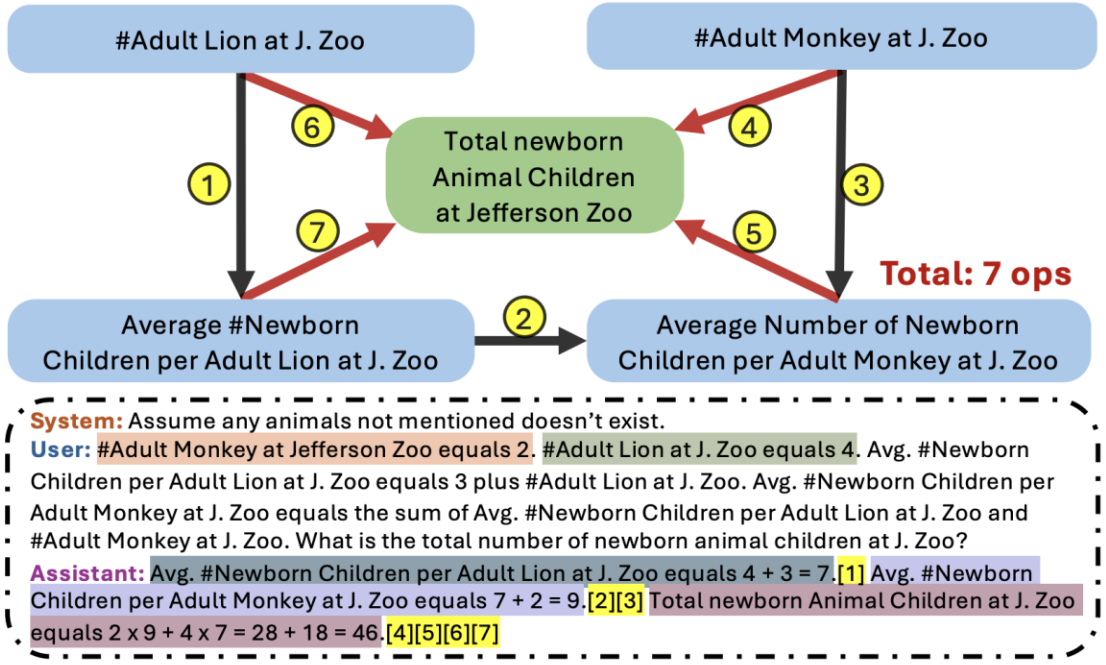




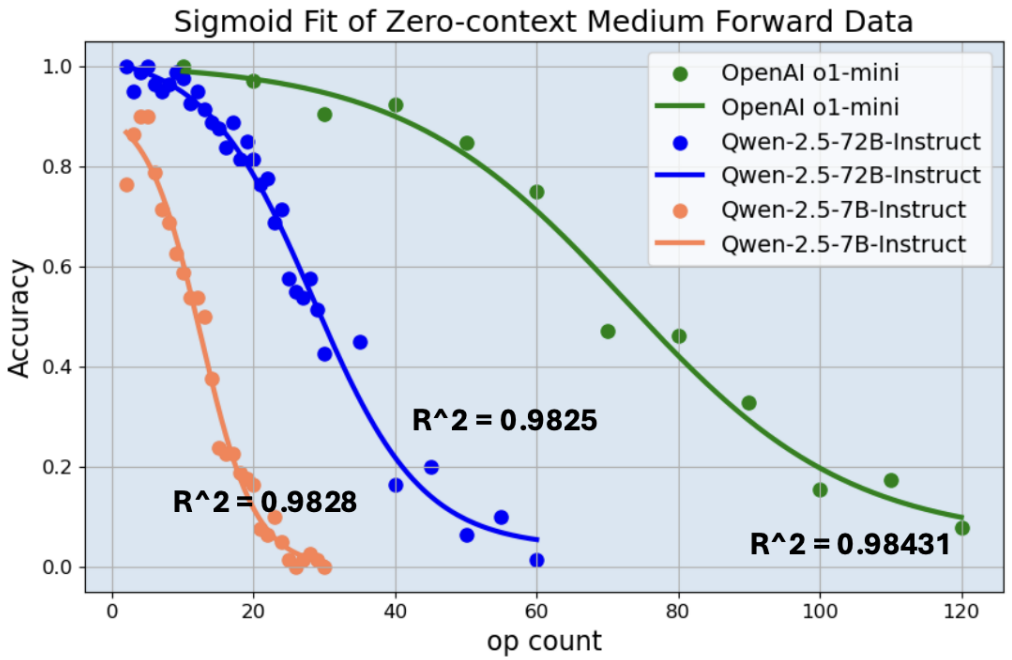


[H. S. Zheng et al, *NATURAL PLAN: Benchmarking LLMs on Natural Language Planning*, arXiv'24]

LLM reasoning & planning is still a hard problem



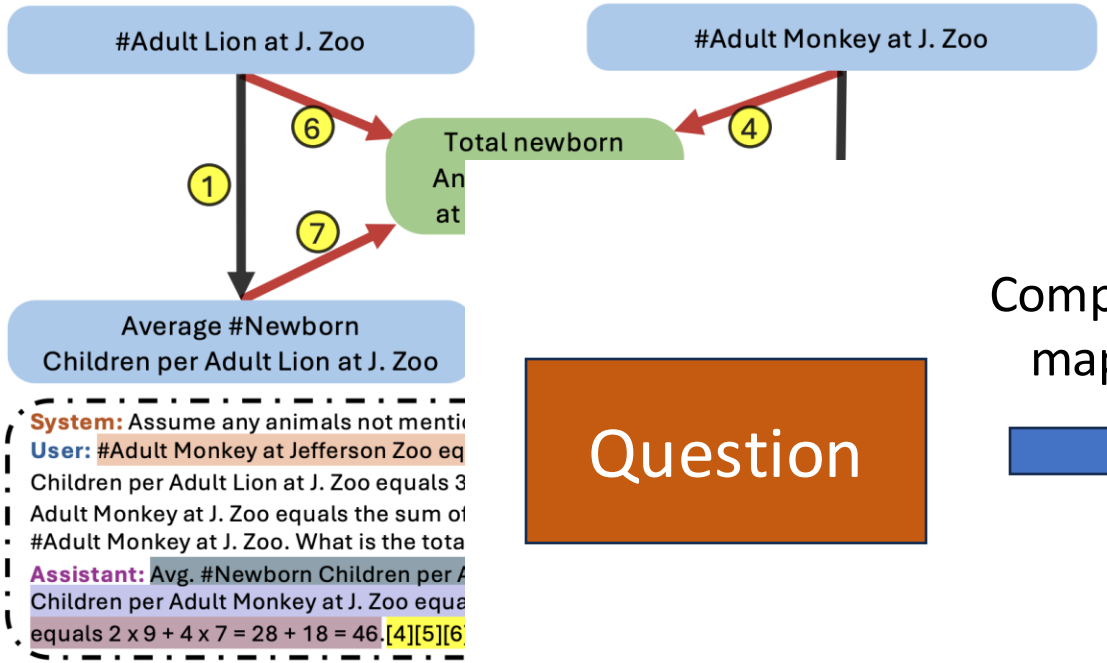
Synthetic Dataset with infinite reasoning complexity



Performance drops with increasing op counts.

 [Y. Zhou et al, *GSM-∞: How Do Your LLMs Behave over Infinitely increasing context length and reasoning complexity?,* arXiv'25]

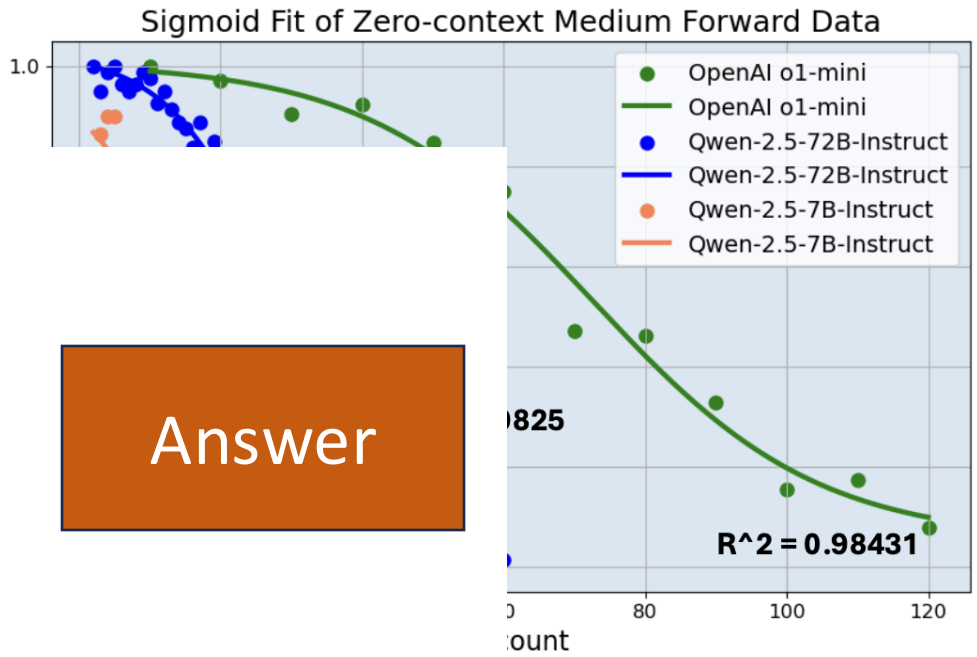
LLM reasoning & planning is still a hard problem



Question

Complicated mapping

Answer



Synthetic Dataset with infinite reasoning complexity

Performance drops with increasing op counts.

 [Y. Zhou et al, *GSM-∞: How Do Your LLMs Behave over Infinitely increasing context length and reasoning complexity?,* arXiv'25]

What are the Solutions?

What are the Solutions?

Option **One**:

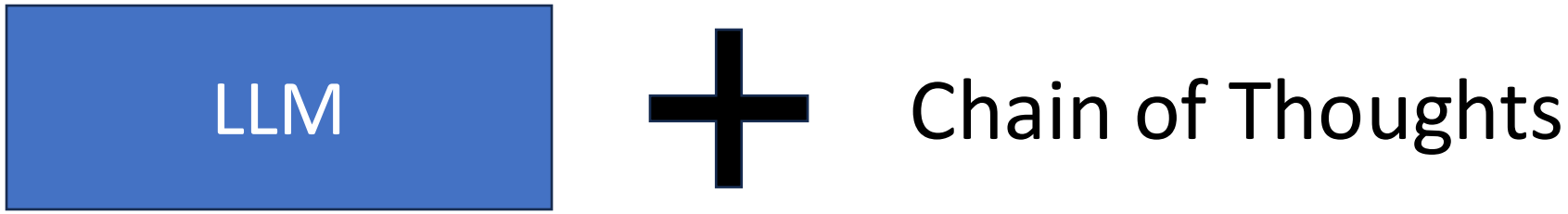
Reasoning by Search

Option **Two**:

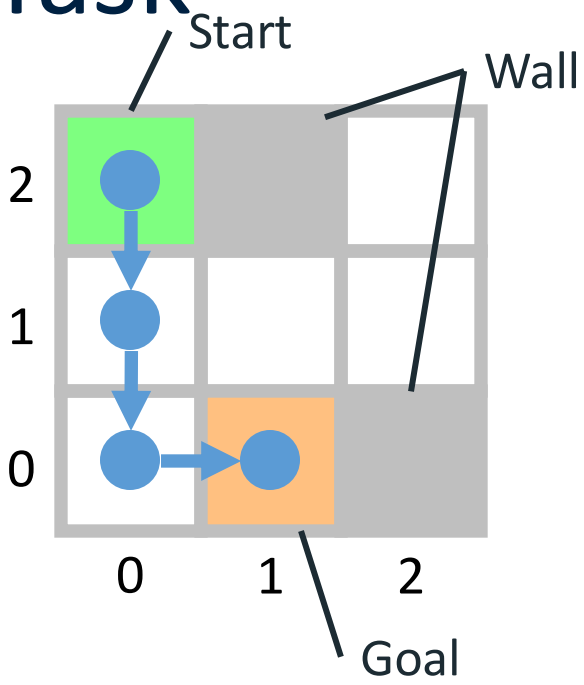
Reasoning by Representation

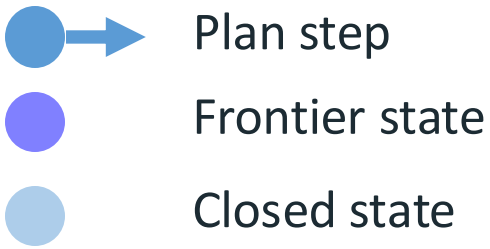
Option **One**: Reasoning by **Search**

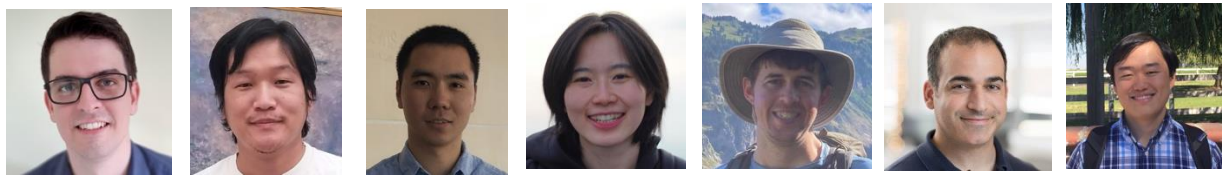
If we cannot get the correct solution right now from LLMs, use more compute to simulate the search behaviors.



Searchformer: A* Search as a Token Prediction Task

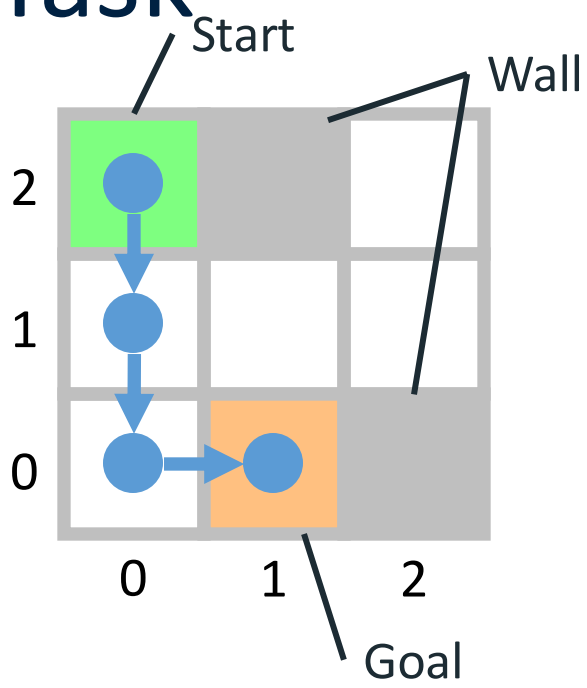






[L. Lehnert, et al, *Beyond A*: Better Planning with Transformers via Search Dynamics Bootstrapping,* COLM'24*]*

Searchformer: A* Search as a Token Prediction Task



Plan step

Frontier state

Closed state

<prompt>

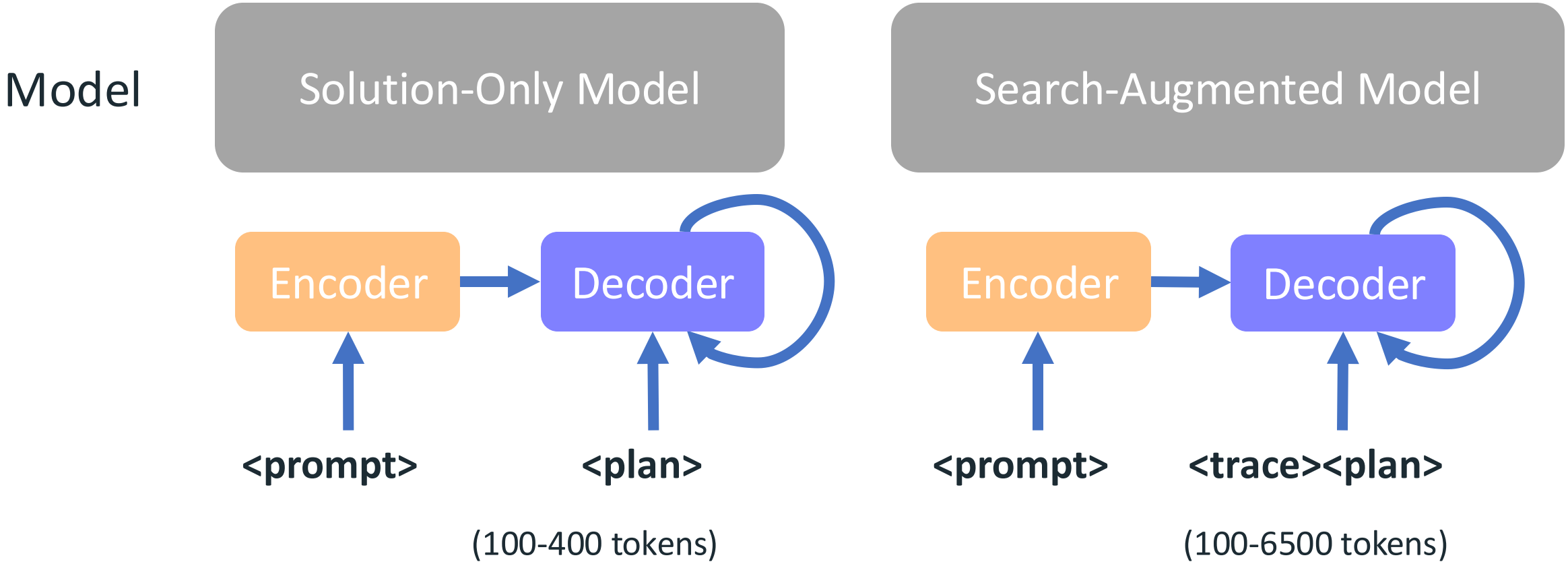
```
bos
start 0 2
goal  1 0
wall  1 2
wall  2 0
eos
```

<trace><plan>

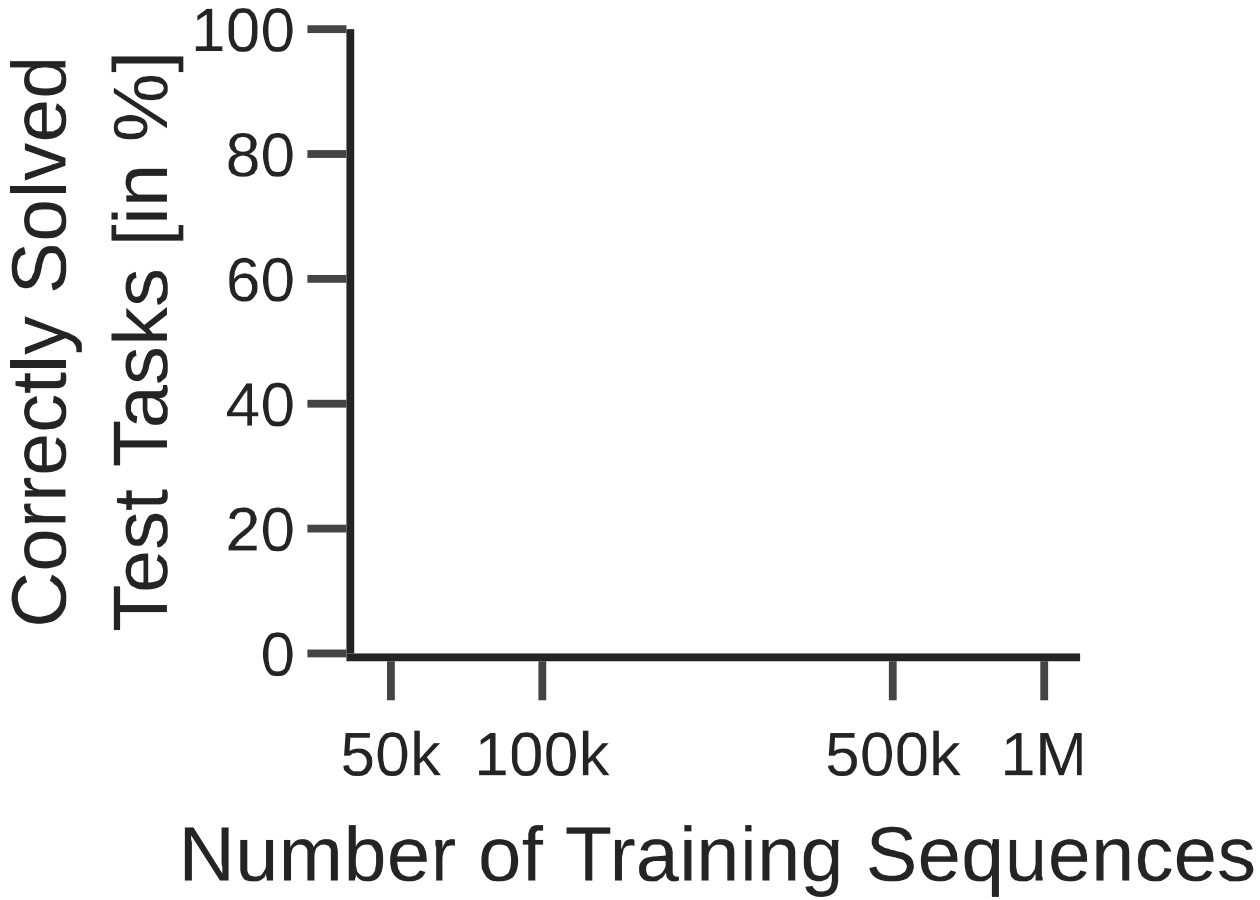
```
bos
create 0 2 c0 c3
close  0 2 c0 c3
create 0 1 c1 c2
close  0 1 c1 c2
create 0 0 c2 c1
create 1 1 c2 c1
close  0 0 c2 c1
create 1 0 c3 c0
close  1 0 c3 c0
plan   0 2
plan   0 1
plan   0 0
plan   1 0
eos
```

Training Method

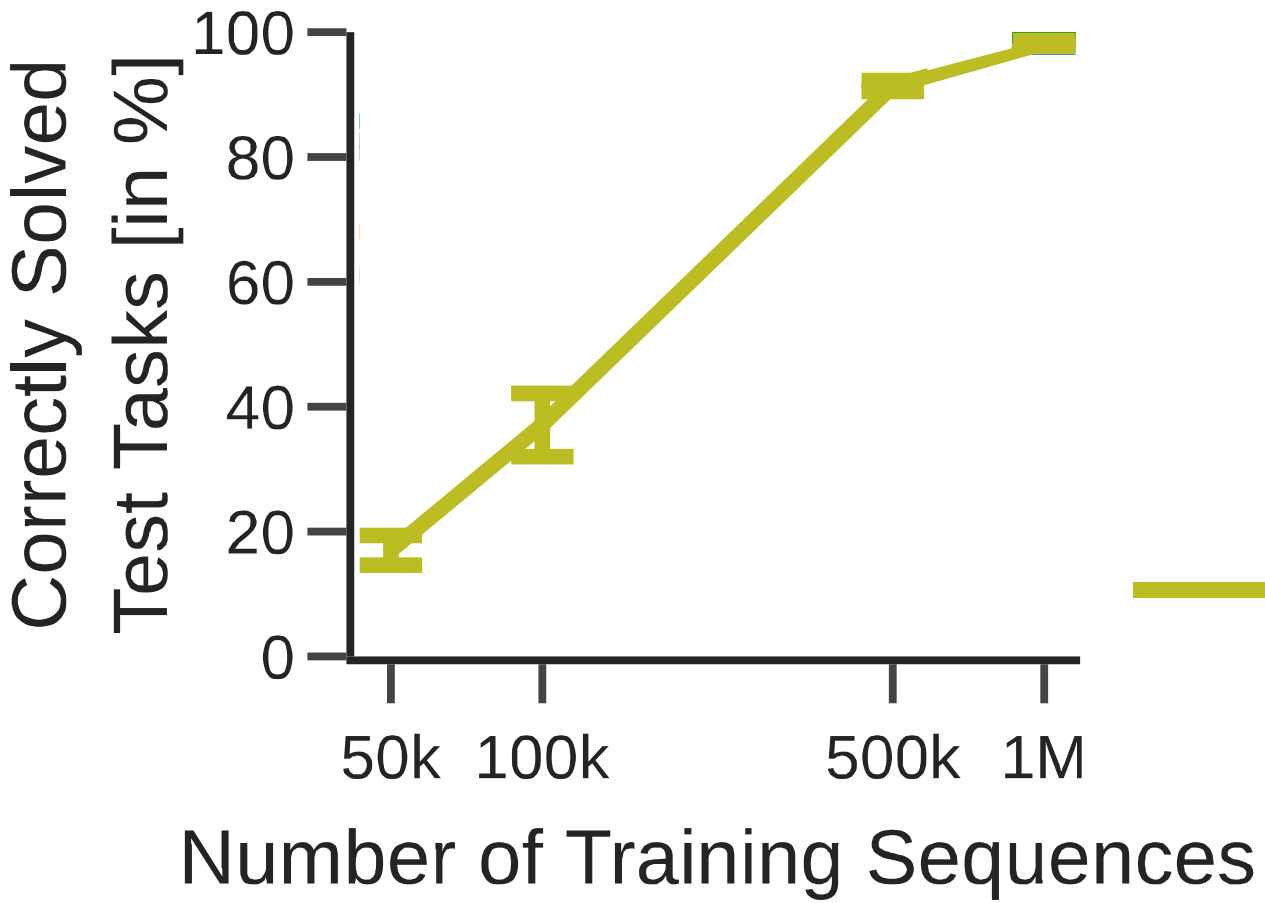
Train a Transformer to predict the next token via teacher forcing.



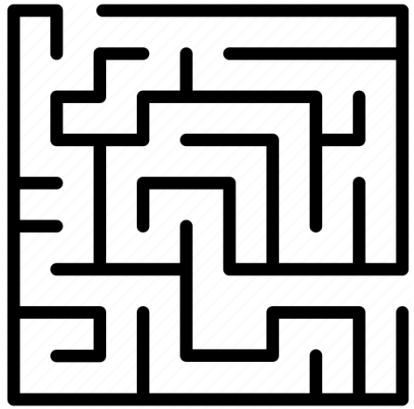
Search-Augmented vs. Solution-Only Models



Search-Augmented vs. Solution-Only Models

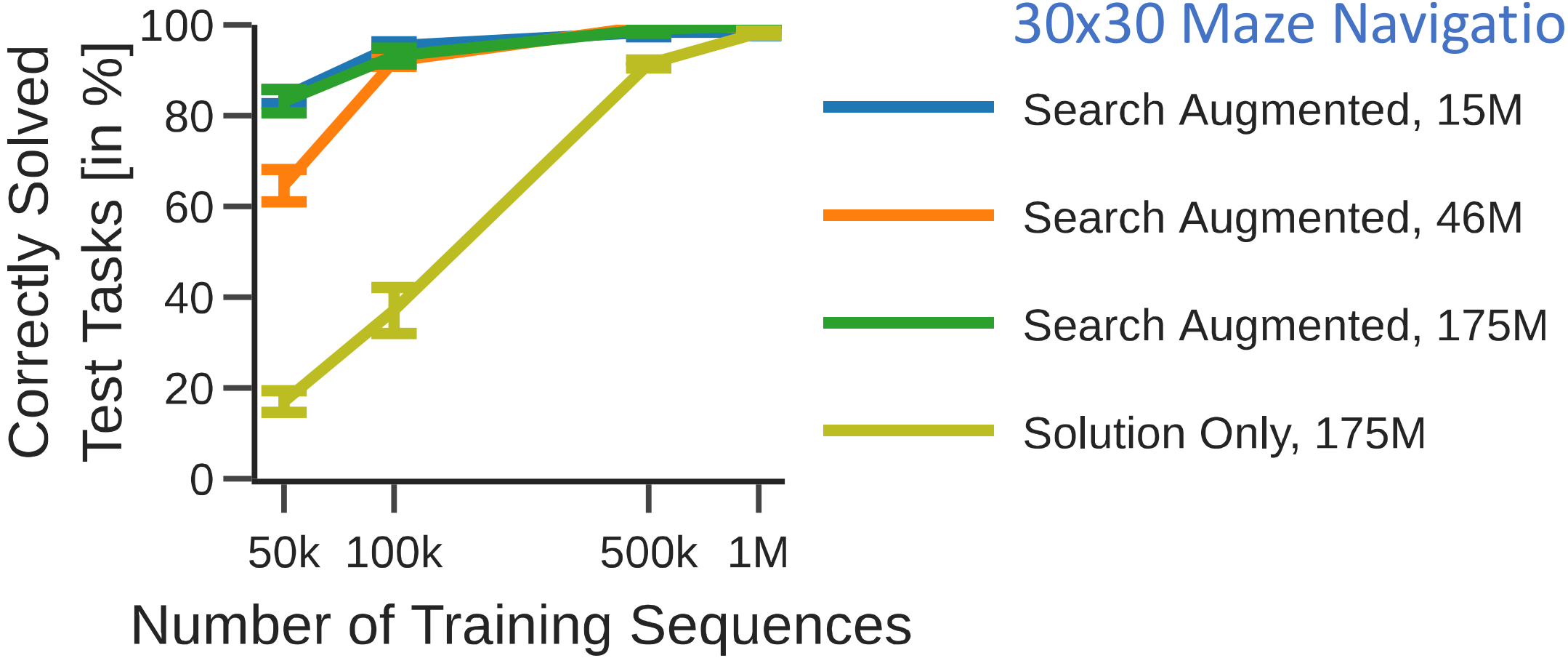


30x30 Maze Navigation

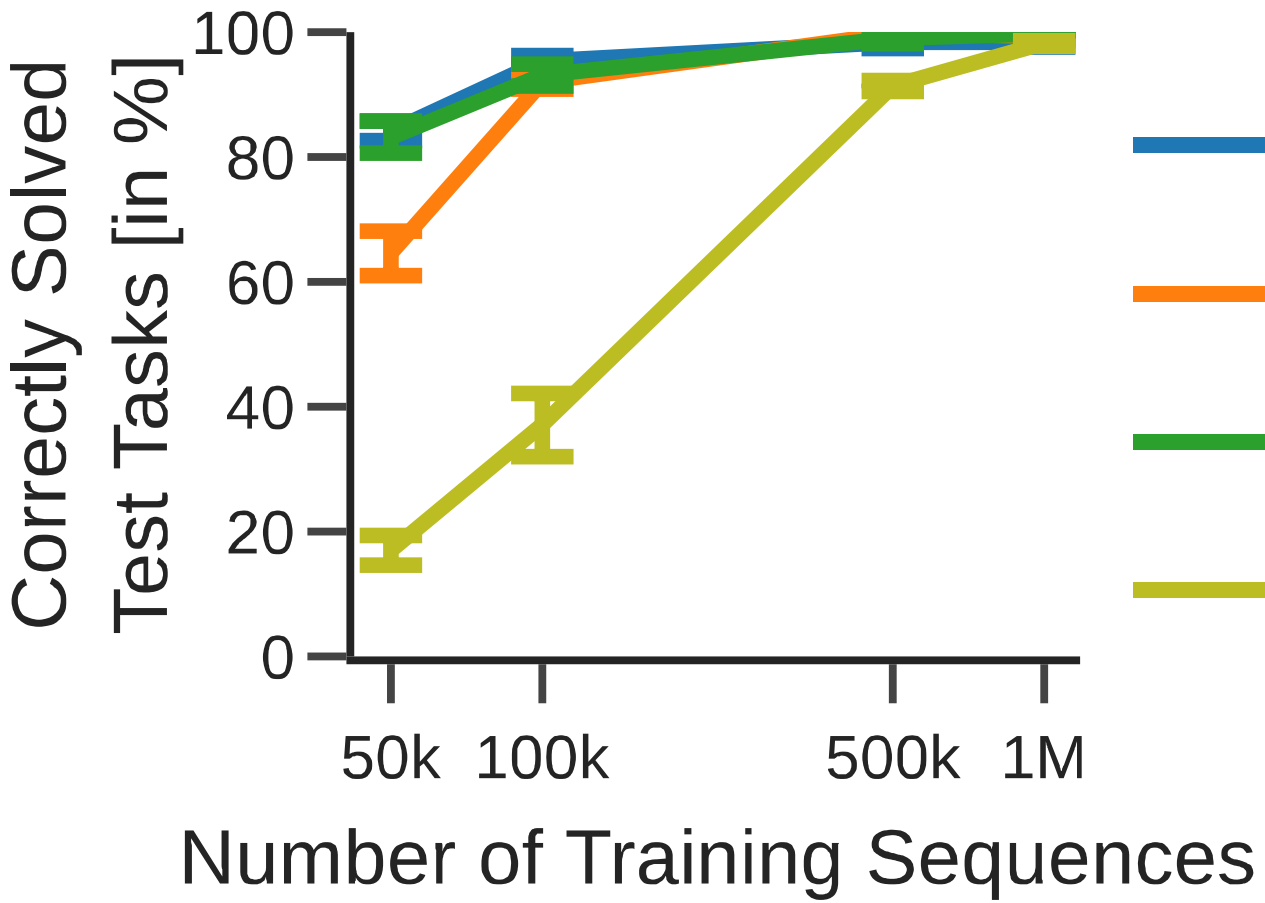


Solution Only, 175M

Search-Augmented vs. Solution-Only Models



Search-Augmented vs. Solution-Only Models

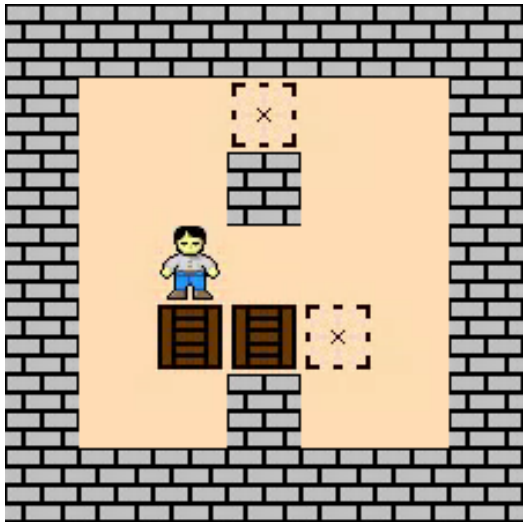


30x30 Maze Navigation

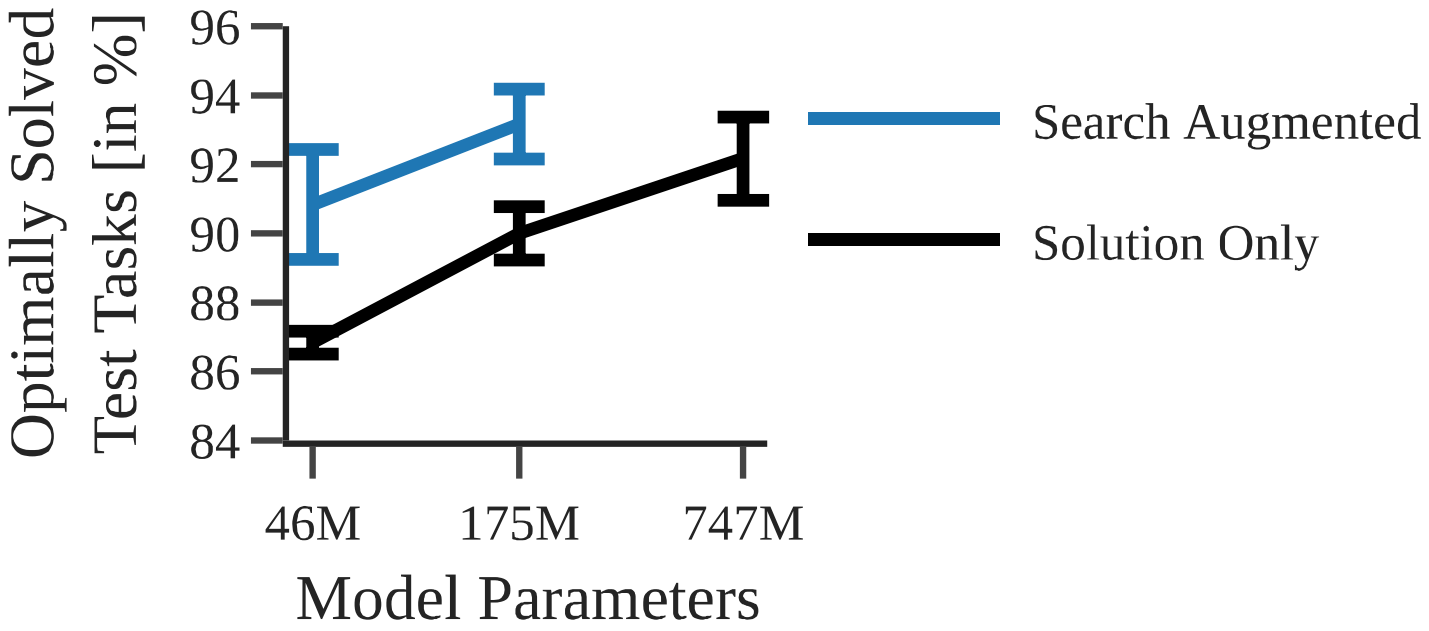
- Search Augmented, 15M
- Search Augmented, 46M
- Search Augmented, 175M
- Solution Only, 175M

Search-augmented is much more parameter & data efficient!

Search-Augmented vs. Solution-Only Models



Sokoban



Search-augmented is much more parameter & data efficient!

How to go beyond?

Imitation Learning

Fine-tuning

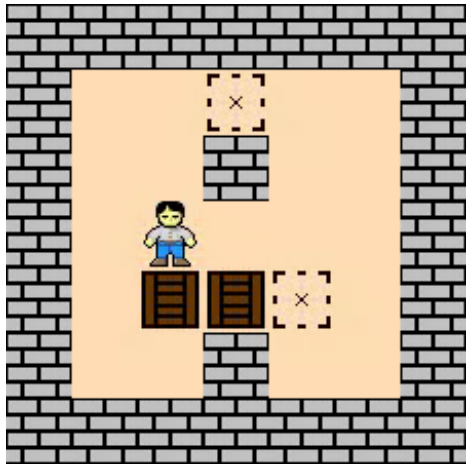
Using solver's trace to train the Transformer with teacher forcing

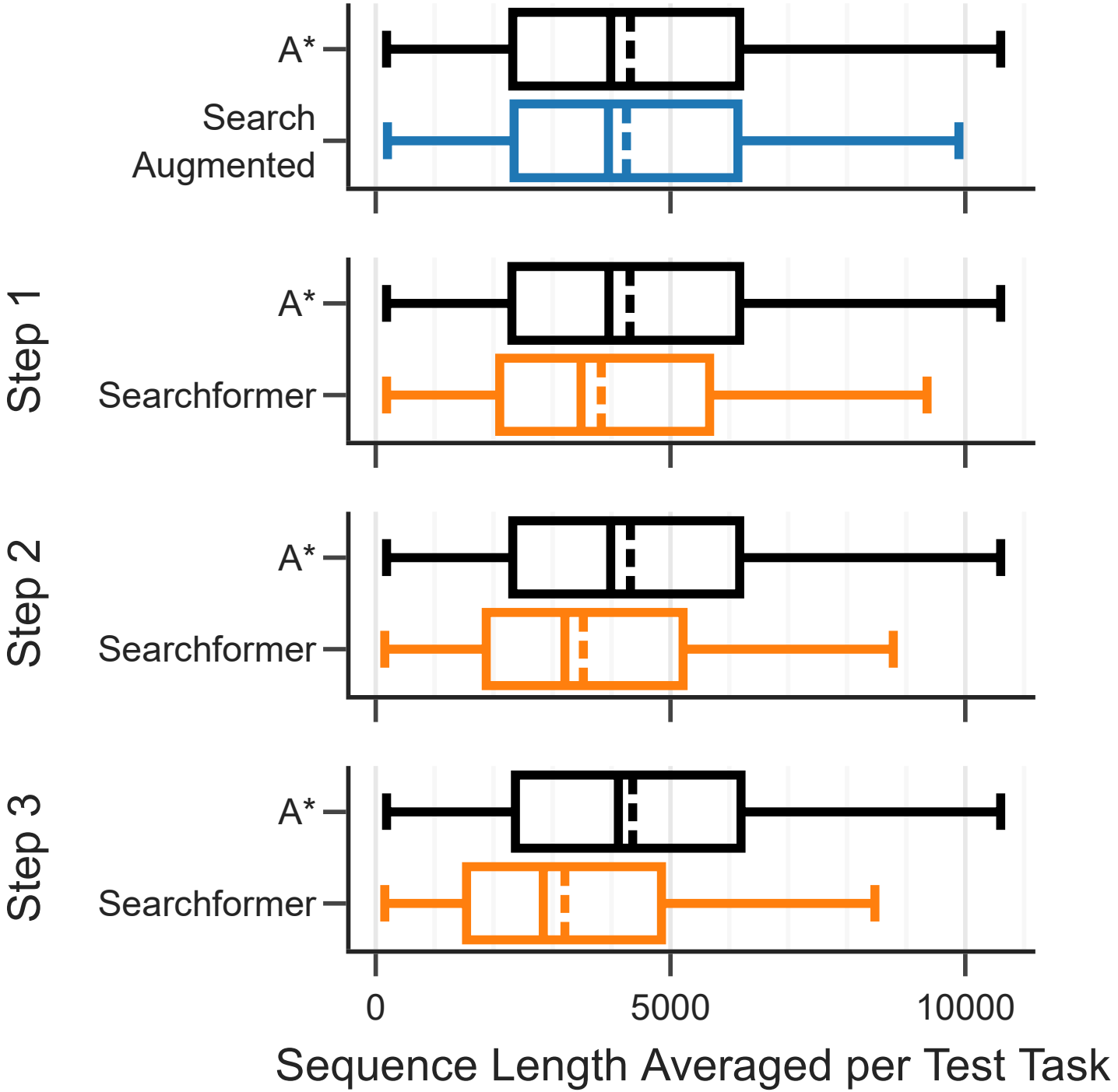
Fine-tune the model to achieve **shorter** trace but still leads to **optimal** plan! (Reinforcement Learning task)

Search-augmented Models

Searchformer

Beyond A*: Improving search dynamics via bootstrapping





Improving search dynamics via bootstrapping

Params.	Model	ILR-on-solved	ILR-on-optimal
45M	Solution only	–	–
	Search augmented	0.908 $\pm$0.020	0.919 $\pm$0.019
	Searchformer, step 1	1.054 $\pm$0.025	1.062 $\pm$0.015
	Searchformer, step 2	1.158 $\pm$0.025	1.181 $\pm$0.012
	Searchformer, step 3	1.292 $\pm$0.044	1.343 $\pm$0.067
175M	Solution only	–	–
	Search augmented	0.925 $\pm$0.010	0.933 $\pm$0.011
757M	Solution only	–	–

Thinking length becomes shorter

$$ILR = \frac{\text{solver len}}{\text{searchformer len}}$$

Improving search dynamics via bootstrapping

Params.	Model	Solved (%)	Optimal (%)
45M	Solution only	90.3 $\pm$1.0	86.8 $\pm$0.3
	Search augmented	92.5 $\pm$1.0	90.8 $\pm$1.6
	Searchformer, step 1	95.5 $\pm$1.0	93.5 $\pm$1.0
	Searchformer, step 2	96.0 $\pm$0.5	93.4 $\pm$0.6
	Searchformer, step 3	95.5 $\pm$0.8	93.7 $\pm$1.6
175M	Solution only	95.7 $\pm$0.2	90.0 $\pm$0.8
	Search augmented	95.2 $\pm$0.9	93.2 $\pm$1.0
757M	Solution only	96.5 $\pm$0.1	92.2 $\pm$1.2

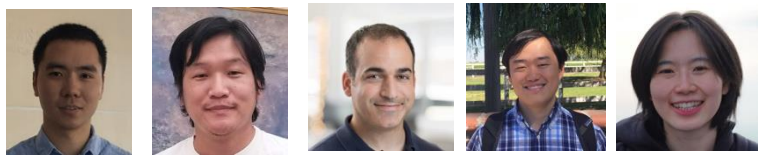
Fine-tuning improves performance initially.

Improving search dynamics via bootstrapping

Params.	Model	Solved (%)	Optimal (%)
45M	Solution only	90.3 ±1.0	86.8 ±0.3
	Search augmented	92.5 ±1.0	90.8 ±1.6
	Searchformer, step 1	95.5 ±1.0	93.5 ±1.0
	Searchformer, step 2	96.0 ±0.5	93.4 ±0.6
	Searchformer, step 3	95.5 ±0.8	93.7 ±1.6
175M	Solution only	95.7 ±0.2	90.0 ±0.8
	Search augmented	95.2 ±0.9	93.2 ±1.0
757M	Solution only	96.5 ±0.1	92.2 ±1.2

Searchformer outperforms largest solution-only model.

DualFormer (Searchformer v2)



A* search Trace

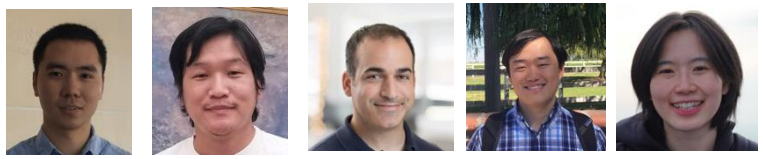
```
bos
create 0 2 c0 c3
close  0 2 c0 c3   <-- D1 (drop a close clause)
create 0 1 c1 c2
close  0 1 c1 c2
create 0 0 c2 c1   <-- D2 (drop cost tokens in a clause)
create 1 1 c2 c1
close  0 0 c2 c1
create 1 0 c3 c0   <-- D3 (drop a create clause)
close  1 0 c3 c0
```

Structured Trace Dropping Strategies

```
LvL 1 = D1            // drop all the close clauses

LvL 2 = D1 + D2       // drop all the close clauses
                      + all the cost tokens

Lvl 3 = D1 + D2 + sampled D3 // Lvl 2 + drop some
                             create clauses

Lvl 4 = drop the entire trace
```


[D. Su et al, *Dualformer: Controllable Fast and Slow Thinking by Learning with Randomized Reasoning Traces*, ICLR'25]

DualFormer (Searchformer v2)



A* search Trace

```
bos
create 0 2 c0 c3
close  0 2 c0 c3   <-- D1 (drop a close clause)
create 0 1 c1 c2
close  0 1 c1 c2
create 0 0 c2 c1   <-- D2 (drop cost tokens in a clause)
create 1 1 c2 c1
close  0 0 c2 c1
create 1 0 c3 c0   <-- D3 (drop a create clause)
close  1 0 c3 c0
```

Structured Trace Dropping Strategies

LvL 1 = D1 // drop all the close clauses

Slow thinking data

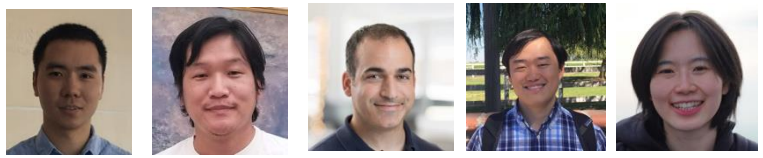
LvL 2 = D1 + D2 // drop all the close clauses + all the cost tokens

Lvl 3 = D1 + D2 + sampled D3 // Lvl 2 + drop some create clauses

Lvl 4 = drop the entire trace

[D. Su et al, *Dualformer: Controllable Fast and Slow Thinking by Learning with Randomized Reasoning Traces*, ICLR'25]

DualFormer (Searchformer v2)



A* search Trace

```
bos
create 0 2 c0 c3
close  0 2 c0 c3   <-- D1 (drop a close clause)
create 0 1 c1 c2
close  0 1 c1 c2
create 0 0 c2 c1   <-- D2 (drop cost tokens in a clause)
create 1 1 c2 c1
close  0 0 c2 c1
create 1 0 c3 c0   <-- D3 (drop a create clause)
close  1 0 c3 c0
```

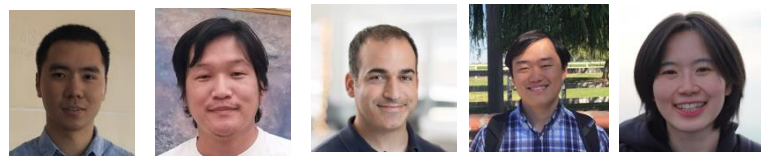
Structured Trace Dropping Strategies

```
LvL 1 = D1                  // drop all the close clauses

LvL 2 = D1 + D2             // drop all the close clauses
                                + all the cost tokens

Lvl 3 = D1 + D2 + sampled D3 // Lvl 2 + drop some
                                create clauses

Lvl 4 = drop the entire trace
```

Fast thinking data

[D. Su et al, *Dualformer: Controllable Fast and Slow Thinking by Learning with Randomized Reasoning Traces*, ICLR'25]

DualFormer (Searchformer v2)



	Method	Avg Trace Length	1-Optimal-64 / 3-Optimal-64	1-Solved-64 / 3-Solved-64	SWC	Diversity
Maze 15 x 15	`Dualformer` (auto)	222	99.7 / 99.4	99.9 / 99.8	0.999	12.52
	Complete-Trace	495	94.6 / 90.1	96.7 / 93.0	0.964	7.60
	Solution-Only	-	72.0 / 68.9	82.7 / 80.1	0.610	1.52
Maze 20 x 20	`Dualformer` (auto)	351	99.5 / 98.6	99.9 / 99.3	0.997	20.28
	Complete-Trace	851	98.3 / 95.5	98.8 / 93.0	0.987	14.53
	Solution-Only	-	56.3 / 52.0	71.9 / 67.5	0.690	1.52
Maze 25 x 25	`Dualformer` (auto)	427	98.6 / 96.9	99.8 / 99.0	0.998	24.81
	Complete-Trace	1208	95.2 / 85.7	97.0 / 90.4	0.968	18.85
	Solution-Only	-	39.7 / 34.7	60.3 / 55.4	0.570	1.9
Maze 30 x 30	`Dualformer` (auto)	617	96.6 / 92.1	98.4 / 97.7	0.989	24.42
	Complete-Trace	1538	93.3 / 82.4	95.9 / 88.1	0.964	7.60
	Solution-Only	-	30.0 / 26.0	54.1 / 47.8	0.500	1.86
Sokoban	`Dualformer` (auto)	494	94.0 / 90.0	97.4 / 94.7	0.979	4.97
	Complete-Trace	3600	92.9 / 84.4	94.7 / 89.0	0.944	2.91
	Solution-Only	-	86.8 / 83.4	92.8 / 90.0	0.919	1.24

Dualformer **automatically** switches between fast mode (System 1) and slow mode (System 2) and works **better** for **dedicated** models on either modes.

Fast mode performance

	Method	1-Optimal-64 / 3-Optimal-64	1-Solved-64 / 3-Solved-64	SWC	Diversity
Maze 15x15	`Dualformer` (fast)	91.8 / 87.6	97.1 / 94.8	0.960	9.05
	Solution-Only	72.0 / 68.9	82.7 / 80.1	0.610	1.52
Maze 20x20	`Dualformer` (fast)	90.9 / 84.0	97.0 / 94.0	0.960	17.27
	Solution-Only	56.3 / 52.0	71.9 / 67.5	0.690	1.52
Maze 25x25	`Dualformer` (fast)	83.9 / 72.9	95.5 / 90.6	0.940	21.23
	Solution-Only	39.7 / 34.7	60.3 / 55.4	0.570	1.9
Maze 30x30	`Dualformer` (fast)	80.0 / 66.0	91.8 / 85.7	0.906	18.23
	Solution-Only	30.0 / 26.0	54.1 / 47.8	0.500	1.86
Sokoban	`Dualformer` (fast)	97.3 / 94.4	94.8 / 90.0	0.970	4.92
	Solution-Only	86.8 / 83.4	92.8 / 90.0	0.919	1.24

Dualformer

Slow thinking

<create>

<plan>

Fast thinking

Slow mode performance

	Method	Avg Trace Length	1-Optimal-64 / 3-Optimal-64	1-Solved-64 / 3-Solved-64	SWC	Diversity
Maze 15 x 15	`Dualformer` (slow)	278	99.6 / 99.2	99.9 / 99.9	0.999	12.54
	Complete-Trace	495	94.6 / 90.1	96.7 / 93.0	0.964	7.60
Maze 20 x 20	`Dualformer` (slow)	439	98.9 / 97.8	99.9 / 99.7	0.998	18.86
	Complete-Trace	851	98.3 / 95.5	98.8 / 93.00	0.987	14.53
Maze 25 x 25	`Dualformer` (slow)	589	99.9 / 97.2	99.7 / 99.3	0.997	25.05
	Complete-Trace	1208	95.2 / 85.7	97.0 / 90.4	0.968	18.85
Maze 30 x 30	`Dualformer` (slow)	854	97.6 / 93.2	99.5 / 98.2	0.993	25.77
	Complete-Trace	1538	93.3 / 82.4	95.9 / 88.1	0.964	7.60
Sokoban	`Dualformer` (slow)	1482	94.5 / 87.6	97.4 / 94.1	0.970	4.66
	Complete-Trace	3600	92.9 / 84.4	94.7 / 89.0	0.944	2.91

Math Problems

Question 1

In Pascal's Triangle, the third row has four elements, and the sum of these elements is 8. What is the sum of the elements in the tenth row? (Note that we are counting the top row with a single 1 as the 0th row).

Llama-3-Aug (baseline) Here is a detailed step-by-step explanation of the reasoning:

- Step 1: Understand the problem and the given information. We are asked to find the sum of the elements in the tenth row of Pascal's Triangle, given that the third row has four elements and their sum is 8.
- Step 2: Recall the formula for the sum of the elements in a row of Pascal's Triangle. The sum of the elements in the nth row is given by the formula: $\binom{n}{0} + \binom{n}{1} + \binom{n}{2} + \cdots + \binom{n}{n} = 2^n$.
- Step 3: Identify the relationship between the sum of the elements in a row and the sum of the elements in the previous row. Since the sum of the elements in the nth row is 2^n, the sum of the elements in the (n-1)th row is 2^{n-1}.
- Step 4: Recognize that the sum of the elements in the third row is 8, which is equal to 2^3. This is because the sum of the elements in the third row is $\binom{3}{0} + \binom{3}{1} + \binom{3}{2} = 1 + 3 + 3 = 7$, and the extra 1 comes from the top row, which is counted as the 0th row.
- Step 5: Use the relationship between the sums of the elements in consecutive rows to find the sum of the elements in the tenth row. Since the sum of the elements in the third row is 2^3, and the sum of the elements in the (n-1)th row is 2^{n-1}, we can set up the equation: $2^3 = 2^{10-1}$.
- Step 6: Solve the equation for n. Taking the logarithm of both sides, we get: $3 \log 2 = (10 - 1) \log 2$. Simplifying, we get: $3 \log 2 = 9 \log 2$, which implies that $\log 2 = 0$. This is a contradiction, since $\log 2$ is a positive number.
- Step 7: Conclude that the equation $2^3 = 2^{10-1}$ has no solution, and therefore, the sum of the elements in the tenth row is not equal to 2^{10-1}.
- Step 8: Recognize that the sum of the elements in the tenth row is still given by the formula: $\binom{10}{0} + \binom{10}{1} + \binom{10}{2} + \cdots + \binom{10}{10} = 2^{10}$.

The answer is: 1024.

Baseline

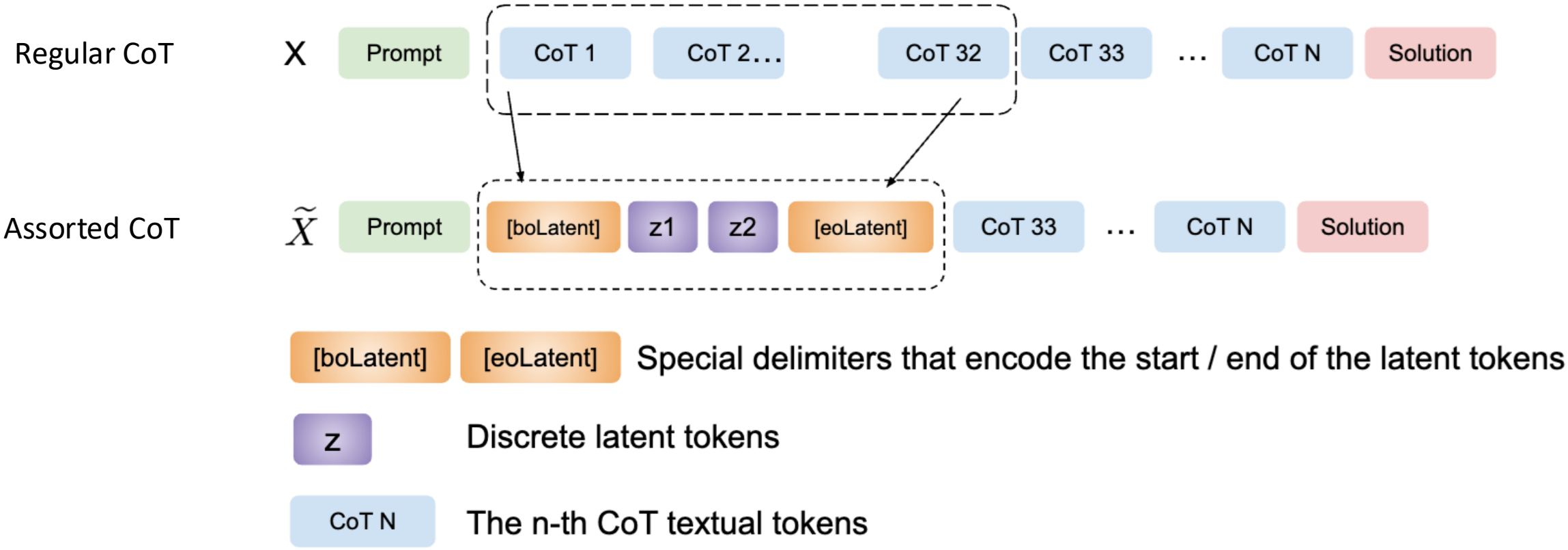
Llama-3-Aug (p=0.4, ours) Here is a detailed step-by-step explanation of the reasoning:

- Step 1: Understand the problem and the given information. We are asked to find the sum of the elements in the tenth row of Pascal's Triangle, given that the third row has four elements and their sum is 8.
- Step 2: Recall the properties of Pascal's Triangle. The elements in each row of Pascal's Triangle are the binomial coefficients, which represent the number of ways to choose k items from a set of n items. The sum of the elements in each row is equal to 2^n, where n is the row number.
- Step 3: Identify the relationship between the sum of the elements in the third row and the sum of the elements in the tenth row. Since the sum of the elements in each row is equal to 2^n, we can write the sum of the elements in the third row as $2^3 = 8$. Similarly, the sum of the elements in the tenth row is 2^{10}.
- Step 4: Calculate the sum of the elements in the tenth row. $2^{10} = 1024$.

The answer is: 1024.

Dualformer

Token Assorted (Searchformer v3)

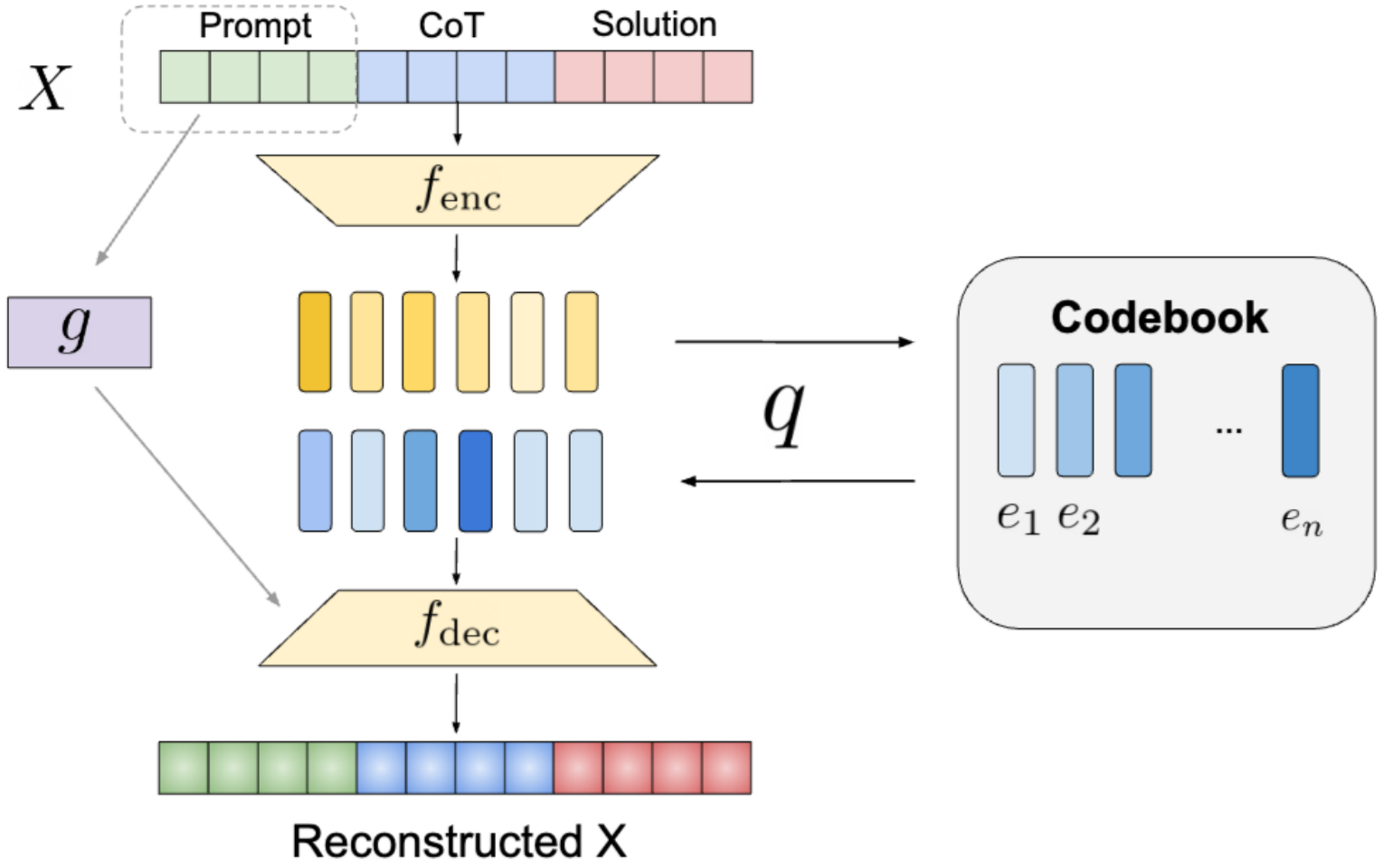


[D. Su et al, *Token Assorted: Mixing Latent and Text Tokens for Improved Language Model Reasoning*, arXiv'25]

Token Assorted (Searchformer v3)

How the latent codes are constructed?

Using VQVAE



Better Performance

Model		In-Domain		Out-of-Domain					Average
		Math	GSM8K	Gaokao-Math-2023	DM-Math	College-Math	Olympia-Math	TheoremQA	All Datasets
Llama-3.2-1B	Sol-Only	4.7	6.8	0.0	10.4	5.3	1.3	3.9	4.6
	CoT	10.5	42.7	**10.0**	3.4	17.1	1.5	9.8	14.1
	iCoT	8.2	10.5	3.3	11.3	7.6	**2.1**	10.7	7.7
	Pause Token	5.1	5.3	2.0	1.4	0.5	0.0	0.6	2.1
	Latent (ours)	**14.7** (↑ +4.2)	**48.7** (↑ +6)	**10.0**	**14.6** (↑ +3.3)	**20.5** (↑ +3.4)	1.8	**11.3** (↑ +0.6)	**17.8** (↑ +3.7)
Llama-3.2-3B	Sol-Only	6.1	8.1	3.3	14.0	7.0	1.8	6.8	6.7
	CoT	21.9	69.7	16.7	**27.3**	30.9	2.2	11.6	25.2
	iCoT	12.6	17.3	3.3	16.0	14.2	**4.9**	**13.9**	11.7
	Pause Token	25.2	53.7	4.1	7.4	11.8	0.7	1.0	14.8
	Latent (ours)	**26.1** (↑ +4.2)	**73.8** (↑ +4.1)	**23.3** (↑ +6.6)	27.1	**32.9** (↑ +2)	4.2	13.5	**28.1** (↑ +2.9)
Llama-3.1-8B	Sol-Only	11.5	11.8	3.3	17.4	13.0	3.8	6.7	9.6
	CoT	32.9	80.1	16.7	39.3	41.9	7.3	15.8	33.4
	iCoT	17.8	29.6	16.7	20.3	21.3	7.6	14.8	18.3
	Pause Token	**39.6**	79.5	6.1	25.4	25.1	1.3	4.0	25.9
	Latent (ours)	37.2	**84.1** (↑ +4.0)	**30.0** (↑ +13.3)	**41.3** (↑ +2)	**44.0** (↑ +2.1)	**10.2** (↑ +2.6)	**18.4** (↑ +2.6)	**37.9** (↑ +4.5)

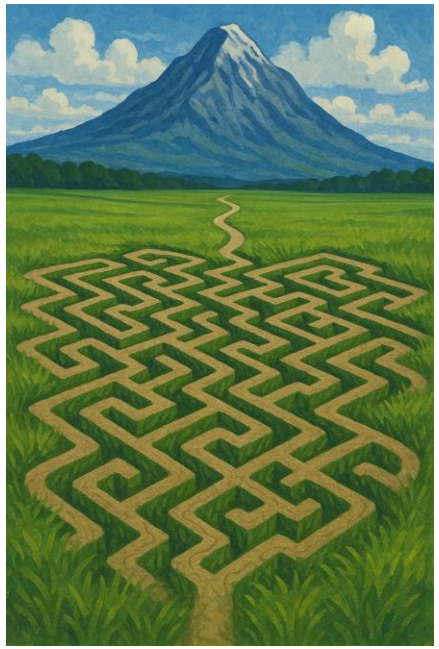
Shorter CoT

Model		In-Domain (# of tokens)		Out-of-Domain (# of tokens)					Average
		Math	GSM8K	Gaokao-Math-2023	DM-Math	College-Math	Olympia-Math	TheoremQA	All Datasets
Llama-3.2-1B	Sol-Only	4.7	6.8	0.0	10.4	5.3	1.3	3.9	4.6
	CoT	646.1	190.3	842.3	578.7	505.6	1087.0	736.5	655.2
	iCoT	328.4	39.8	354.0	170.8	278.7	839.4	575.4	369.5
	Pause Token	638.8	176.4	416.1	579.9	193.8	471.9	988.1	495
	Latent (ours)	501.6 (↓ **-22%**)	181.3 (↓ **-5%**)	760.5 (↓ **-11%**)	380.1 (↓ **-34%**)	387.3 (↓ **-23%**)	840.0 (↓ **-22%**)	575.5 (↓ **-22%**)	518 (↓ **-21%**)
Llama-3.2-3B	Sol-Only	6.1	8.1	3.3	14.0	7.0	1.8	6.8	6.7
	CoT	649.9	212.1	823.3	392.8	495.9	1166.7	759.6	642.9
	iCoT	344.4	60.7	564.0	154.3	224.9	697.6	363.6	344.2
	Pause Token	307.9	162.3	108.9	251.5	500.96	959.5	212.8	354.7
	Latent (ours)	516.7 (↓ **-20%**)	198.8 (↓ **-6%**)	618.5 (↓ **-25%**)	340.0 (↓ **-13%**)	418.0 (↓ **-16%**)	832.8 (↓ **-29%**)	670.2 (↓ **-12%**)	513.6 (↓ **-20%**)
Llama-3.1-8B	Sol-Only	11.5	11.8	3.3	17.4	13.0	3.8	6.7	9.6
	CoT	624.3	209.5	555.9	321.8	474.3	1103.3	760.1	578.5
	iCoT	403.5	67.3	444.8	137.0	257.1	797.1	430.9	362.5
	Pause Token	469.4	119.0	752.6	413.4	357.3	648.2	600.1	480
	Latent (ours)	571.9 (↓ **-9 %**)	193.9 (↓ **-8 %**)	545.8 (↓ **-2 %**)	292.1 (↓ **-10%**)	440.3 (↓ **-8%**)	913.7 (↓ **-17 %**)	637.2 (↓ **-16 %**)	513.7 (↓ **-10%**)

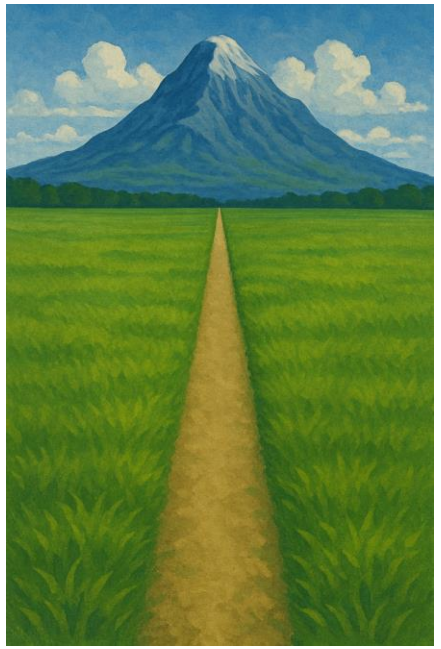
What's wrong with Option **One**?

If search can solve the problem, why not using traditional symbolic solvers?

Option **Two**: Reasoning by **Representation**



Complicated search if we **don't understand** the problem



Simple and straightforward to the destination if we **understand** the problem

Instead of reasoning **exhaustively**, we reason **smartly.**

Option **Two**: Reasoning by **Representation**

Representation

- (Traditional) Symbolic representation

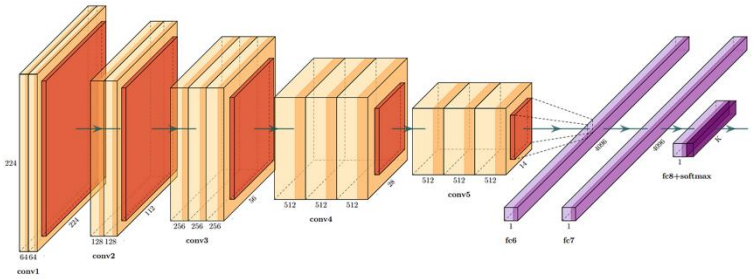
$$\nabla \cdot \mathbf{E} = \frac{\rho_v}{\varepsilon} \quad \text{(Gauss' Law)}$$

$$\nabla \cdot \mathbf{H} = 0 \quad \text{(Gauss' Law for Magnetism)}$$

$$\nabla \times \mathbf{E} = -\mu \frac{\partial \mathbf{H}}{\partial t} \quad \text{(Faraday's Law)}$$

$$\nabla \times \mathbf{H} = \mathbf{J} + \varepsilon \frac{\partial \mathbf{E}}{\partial t} \quad \text{(Ampere's Law)}$$

- Neural Representation



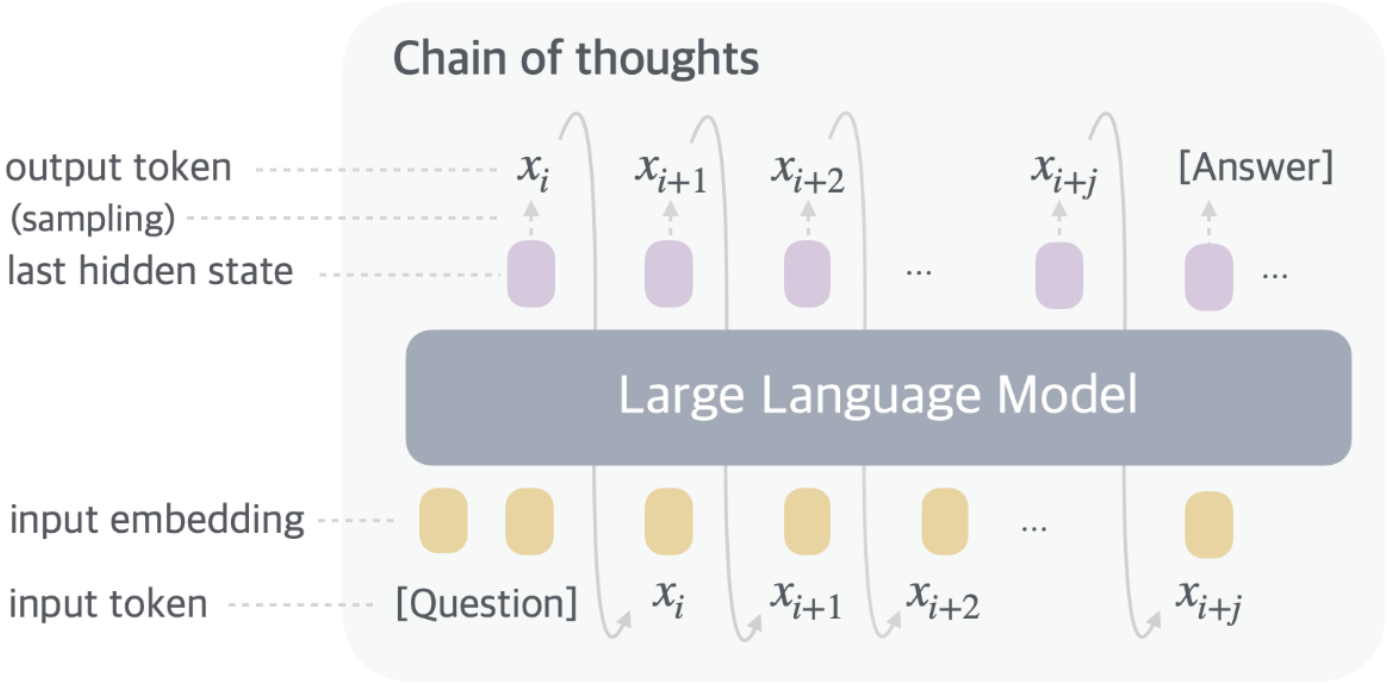
Option **Two**: Reasoning by **Representation**

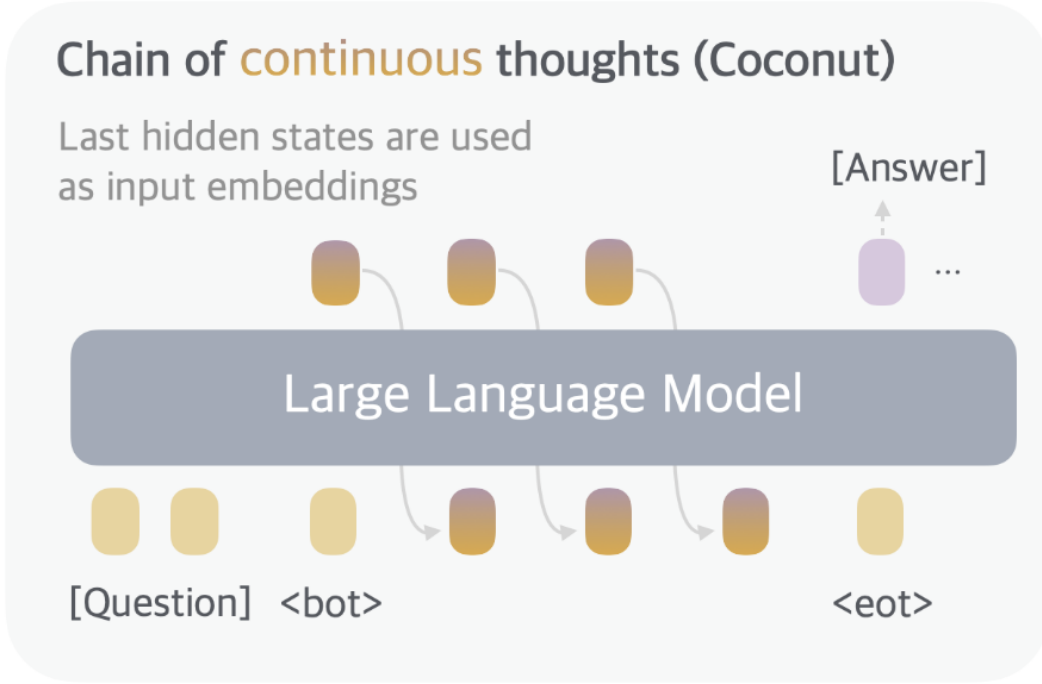
Representation

- (Traditional) Symbolic representation
- Neural Representation → 

Emerging Symbolic Structure

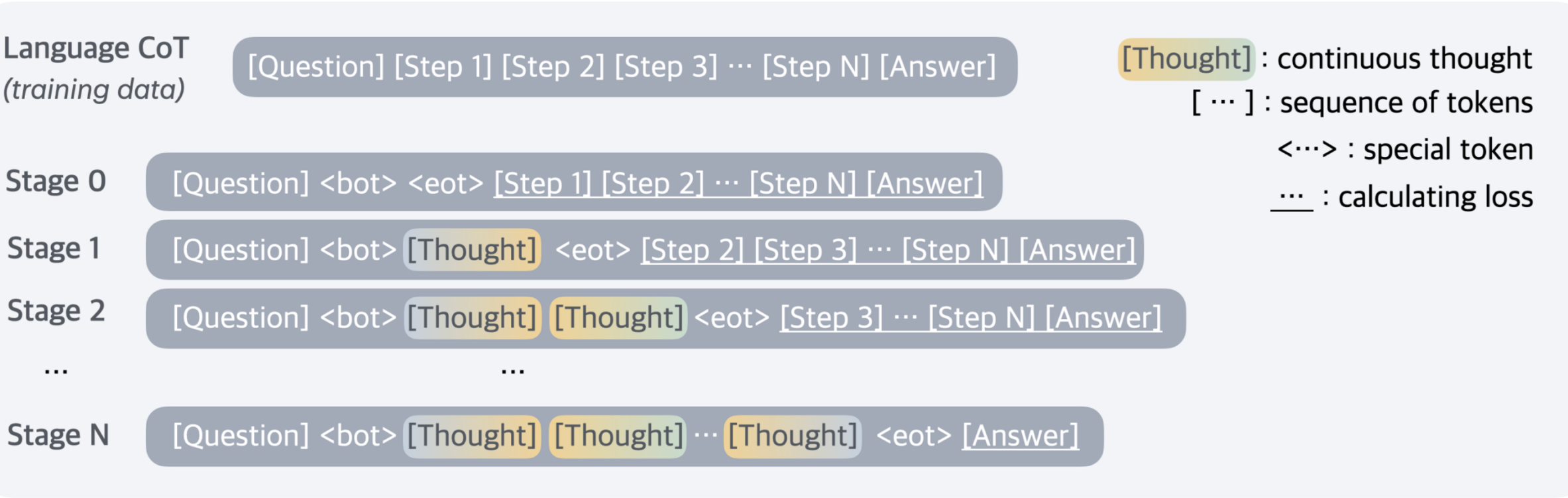
CoConut (Chain of Continuous Thought)



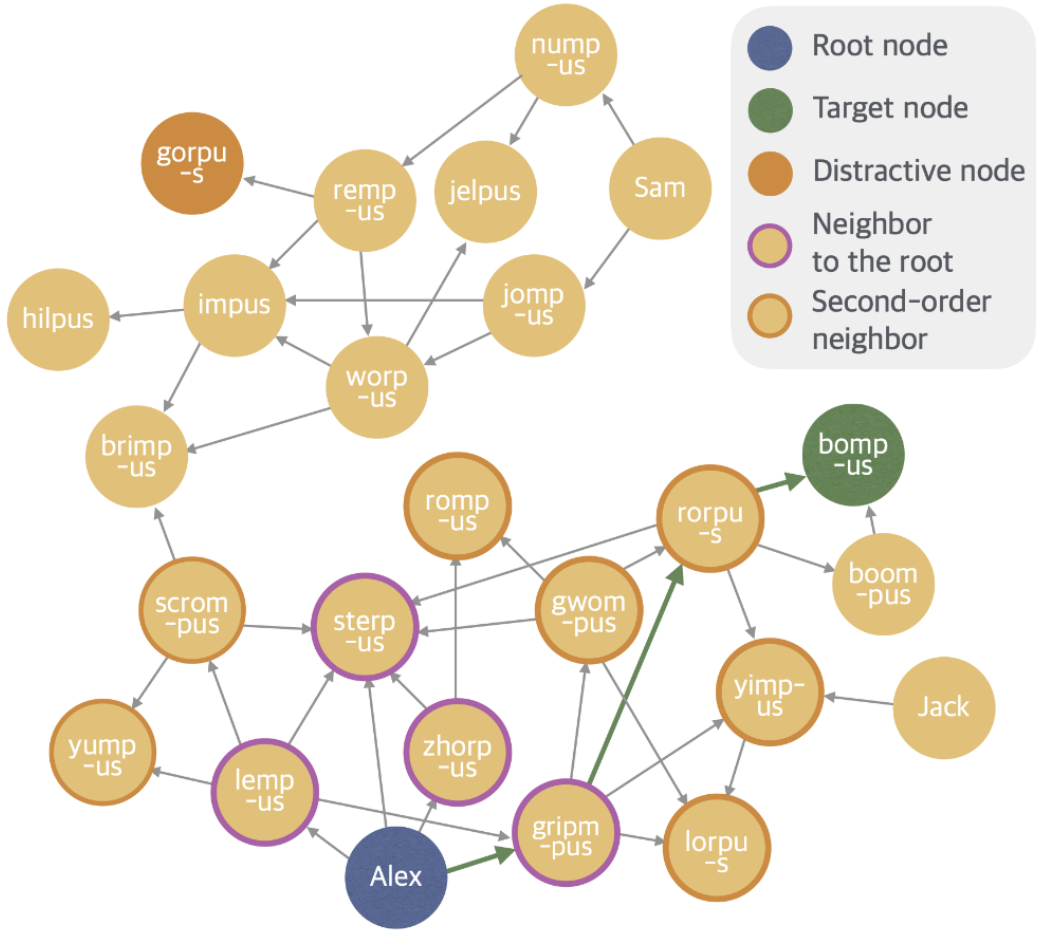


[S. Hao et al, *Training Large Language Models to Reason in a Continuous Latent Space*, arXiv'24]

How to train Coconut?



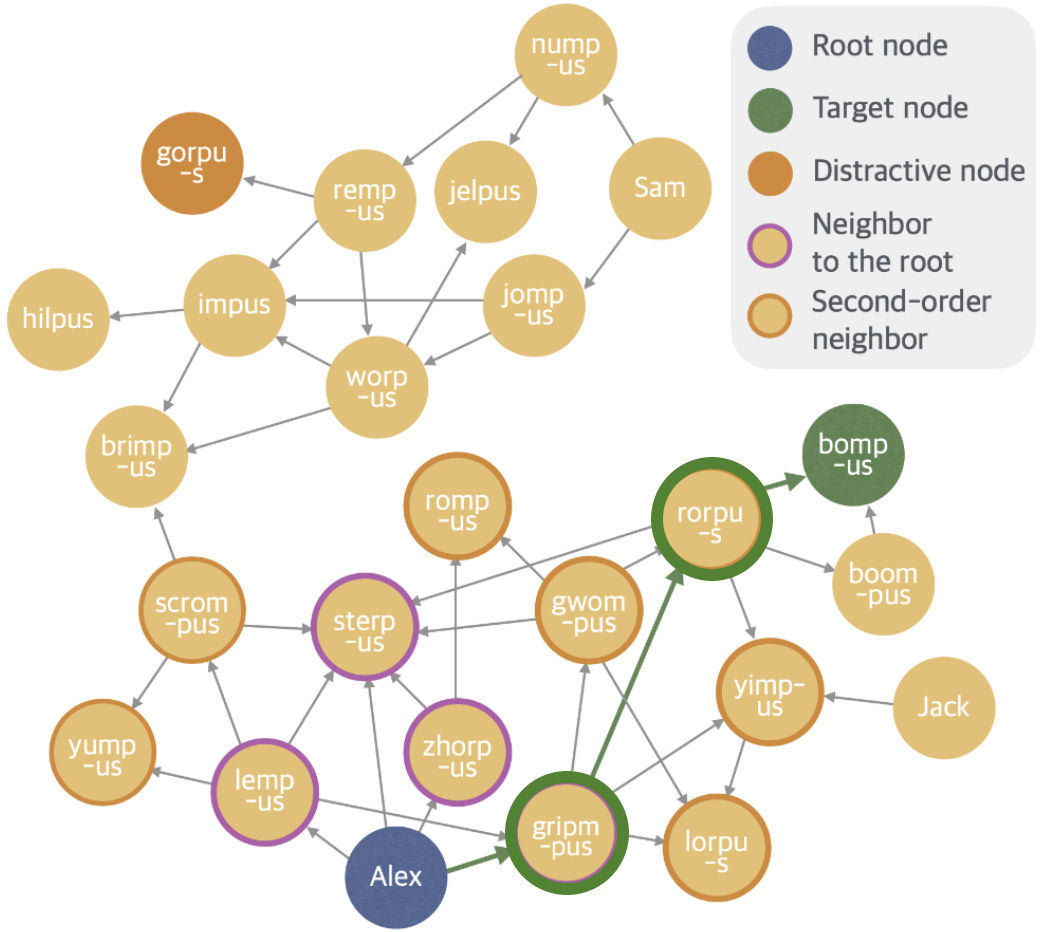
Interpreting the embeddings



Question:

Every jells is a worpus. Sam is a jumpus. Every gwompus is a rompus. ⋯ Every lumps is a yumpus. Question: **Is Alex a gorpus or bompus?**

Ground Truth Solutions



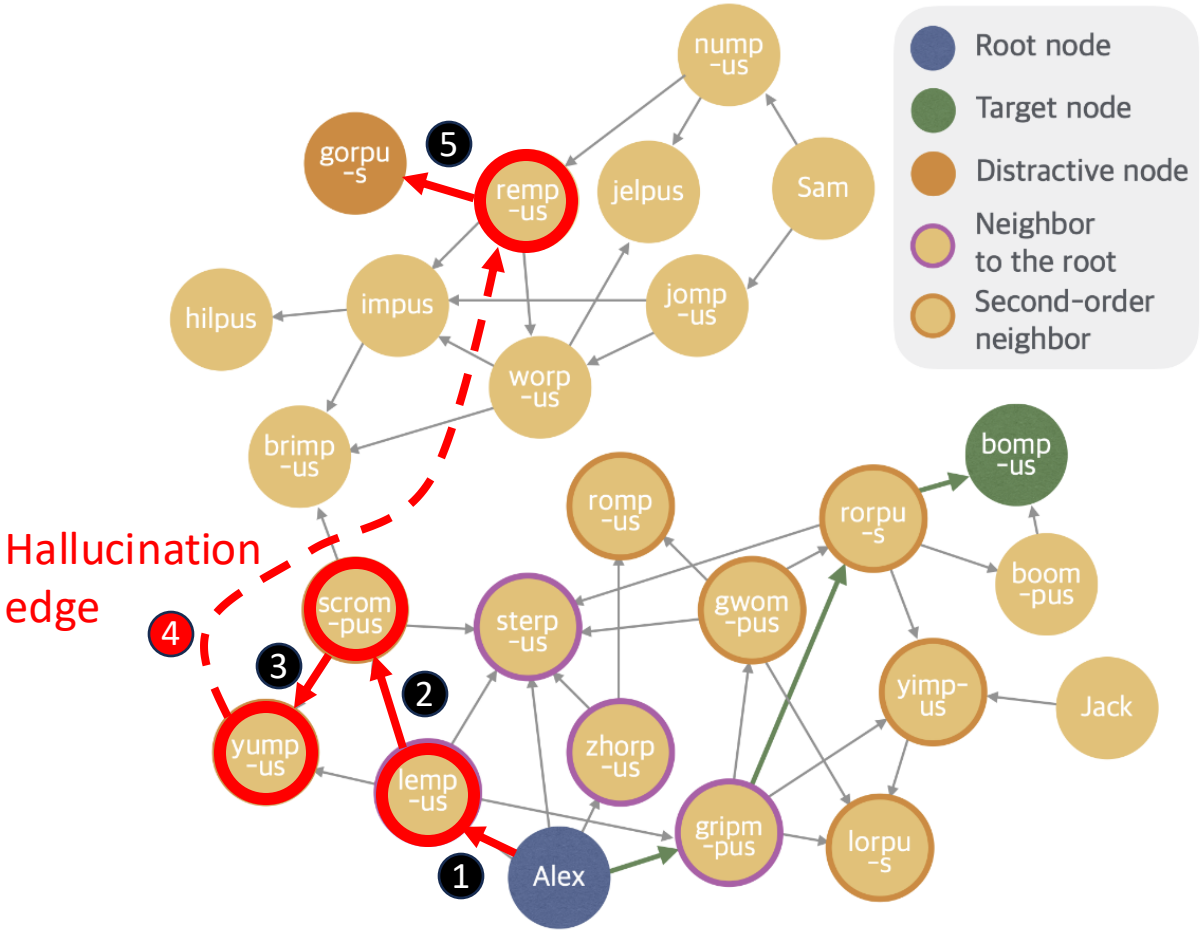
Question:

Every jells is a worpus. Sam is a jumpus. Every gwompus is a rompus. ⋯ Every lumps is a yumpus. Question: **Is Alex a gorpus or bompus?**

Ground Truth Solution

Alex is a grimpus.
Every grimpus is a rorpus.
Every rorpus is a bompus.
Alex is a bompus

Chain of thoughts lead to hallucinations



Question:

Every jells is a worpus. Sam is a jumpus. Every gwompus is a rompus. ⋯ Every lumps is a yumpus. Question: **Is Alex a gorpus or bompus?**

Ground Truth Solution

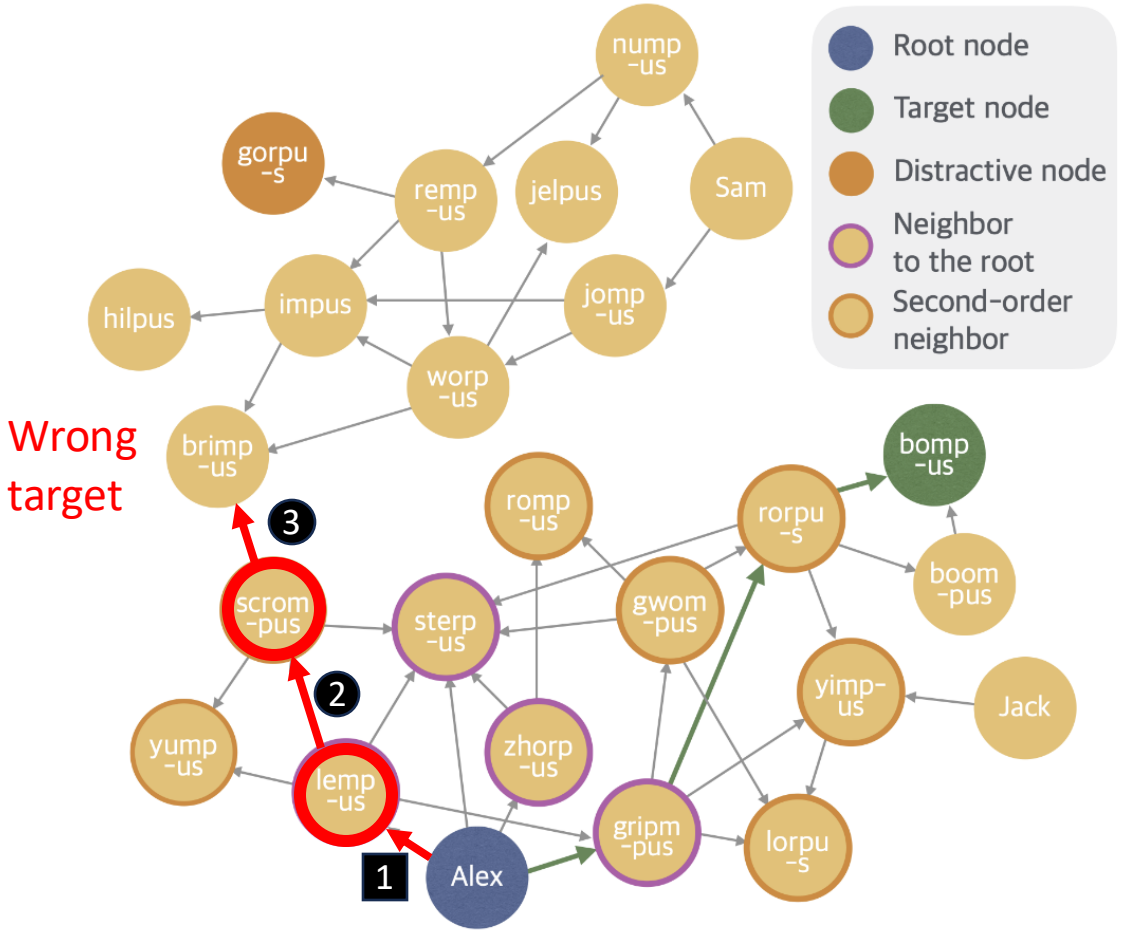
Alex is a grimpus.
Every grimpus is a rorpus.
Every rorpus is a bompus.
Alex is a bompus

CoT

Alex is a lempus. 1
Every lempus is a scrompus. 2
Every scrompus is a yumpus. 3
Every yumpus is a rempus. 4
Every rempus is a gorpus. 5
Alex is a gorpus ✗

(Hallucination)

Continuous Thoughts



Question:

Every jells is a worpus. Sam is a jumpus. Every gwompus is a rompus. ⋯ Every lumps is a yumpus. Question: **Is Alex a gorpus or bompus?**

Ground Truth Solution

Alex is a grimpus.
Every grimpus is a rorpus.
Every rorpus is a bompus.
Alex is a bompus

CoT

Alex is a lempus.
Every lempus is a scrompus.
Every scrompus is a yumpus.
Every yumpus is a rempus.
Every rempus is a gorpus.
Alex is a gorpus ✗

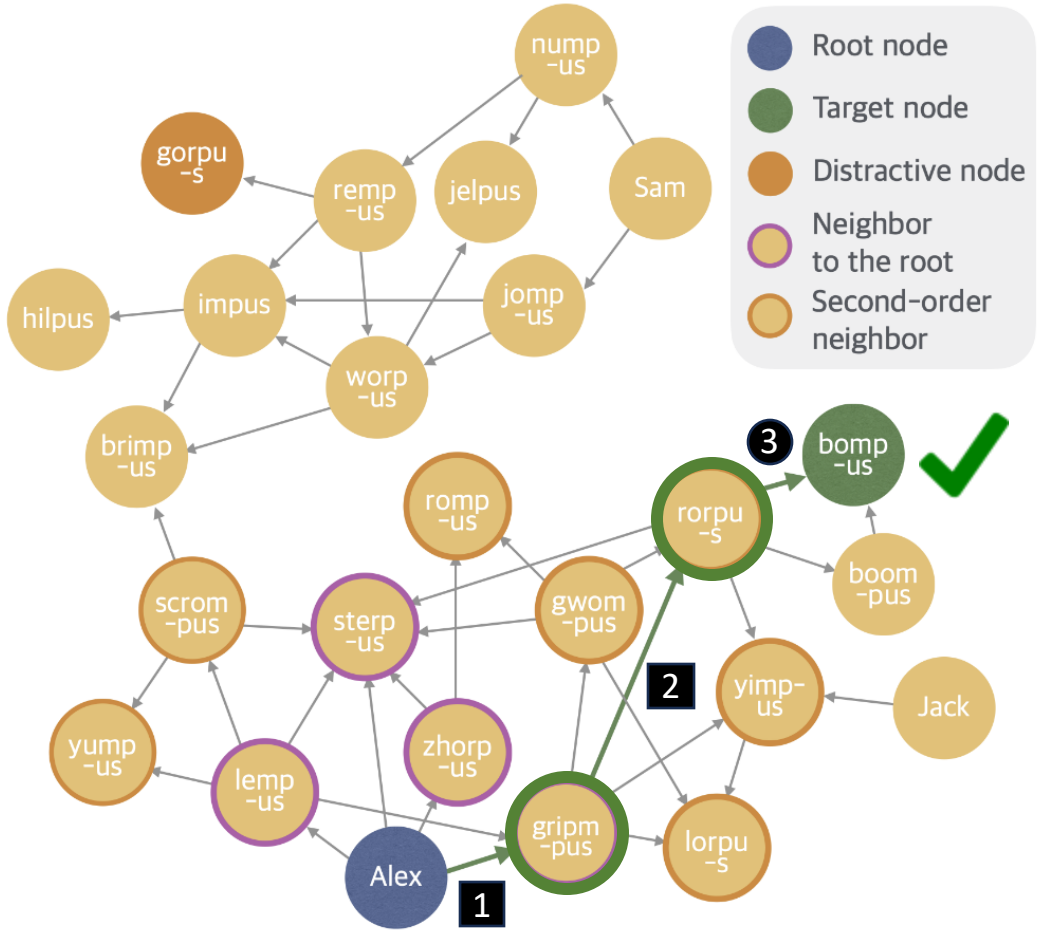
(Hallucination)

Ours (k=1)

<bot> [Thought] <eot> 1
Every lempus is a scrompus. 2
Every scrompus is a brimpus. 3
Alex is a brimpus ✗

(Wrong Target)

Two-step Continuous Thought works!



Question:

Every jells is a worpus. Sam is a jumpus. Every gwompus is a rompus. ⋯ Every lumps is a yumpus. Question: **Is Alex a gorpus or bompus?**

Ground Truth Solution

Alex is a grimpus.
Every grimpus is a rorpus.
Every rorpus is a bompus.
Alex is a bompus

CoT

Alex is a lempus.
Every lempus is a scrompus.
Every scrompus is a yumpus.
Every yumpus is a rempus.
Every rempus is a gorpus.
Alex is a gorpus ✗

(Hallucination)

Ours (k=1)

<bot> [Thought] <eot>
Every lempus is a scrompus.
Every scrompus is a brimpus.
Alex is a brimpus ✗

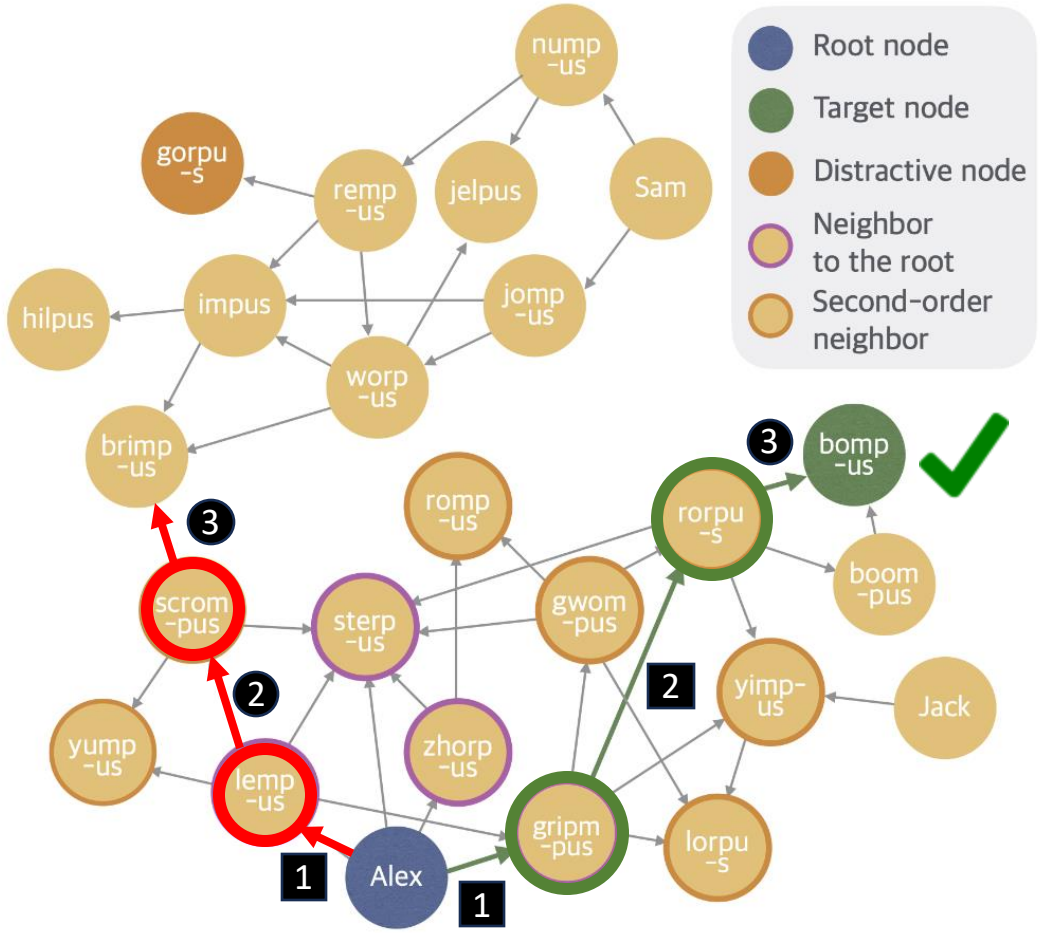
(Wrong Target)

Ours (k=2)

<bot> [thought] [thought] <eot>
Every rorpus is a bompus. 3
Alex is a bompus ✓

(Correct Path)

Two-step Continuous Thought works!



Ours (k=1) 1 2 3

<bot> [Thought] <eot>
Every lempus is a scrompus.
Every scrompus is a brimpus.
Alex is a brimpus ✗

(Wrong Target)

Ours (k=2) 1 2 3

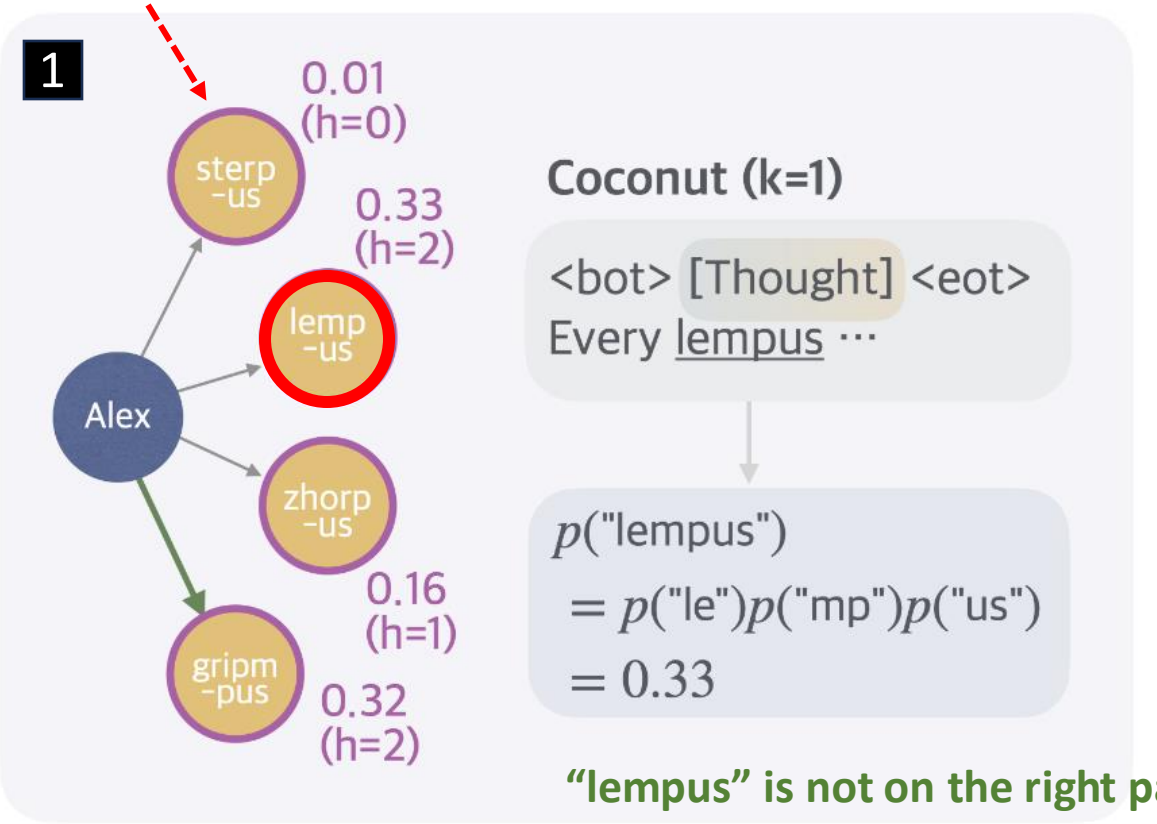
<bot> [thought] [thought] <eot>
Every rorpus is a bompus.
Alex is a bompus ✓

(Correct Path)

Why the same continuous thoughts 1 lead to different path?!

What's inside?

Let's probe!

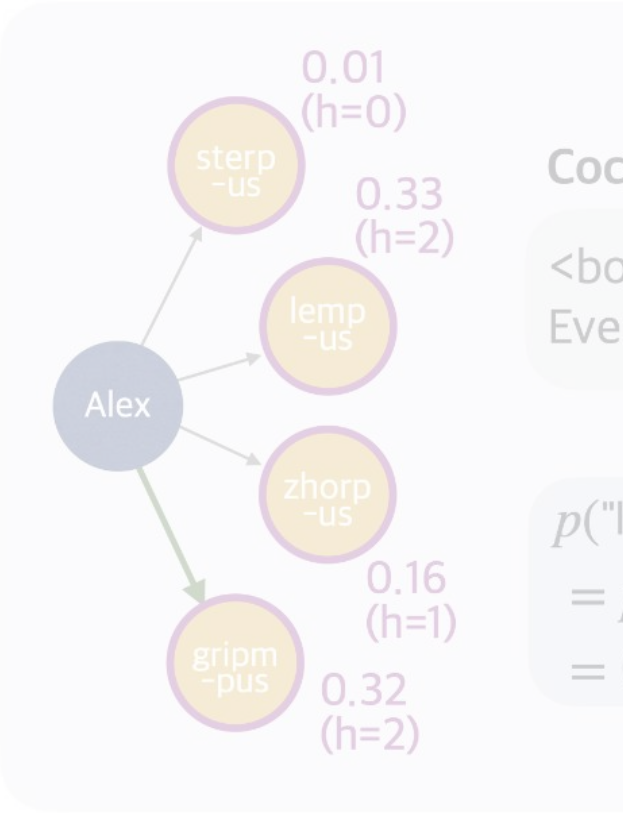


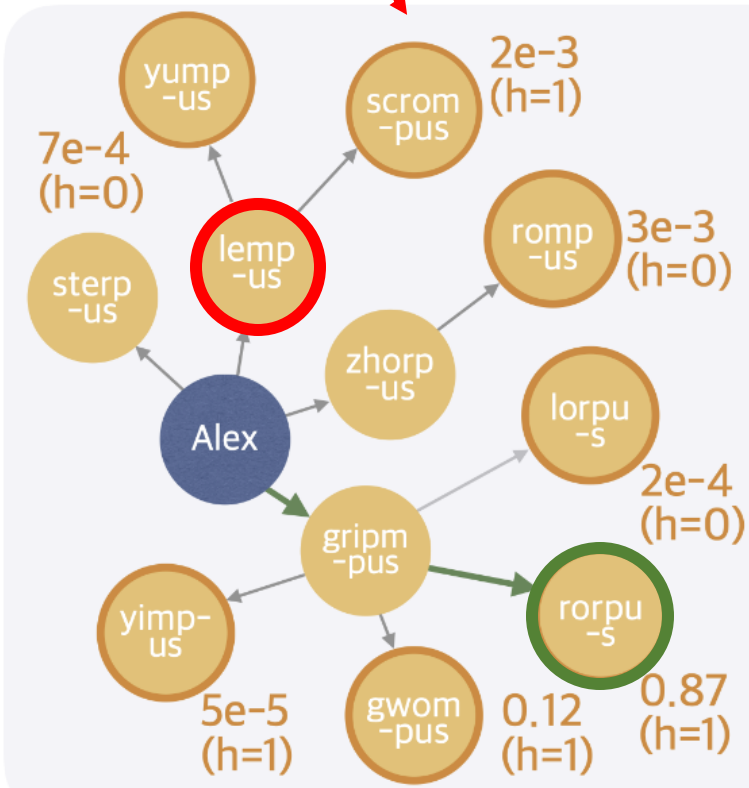
"lempus" is not on the right path but for step=1, it is the **most promising**

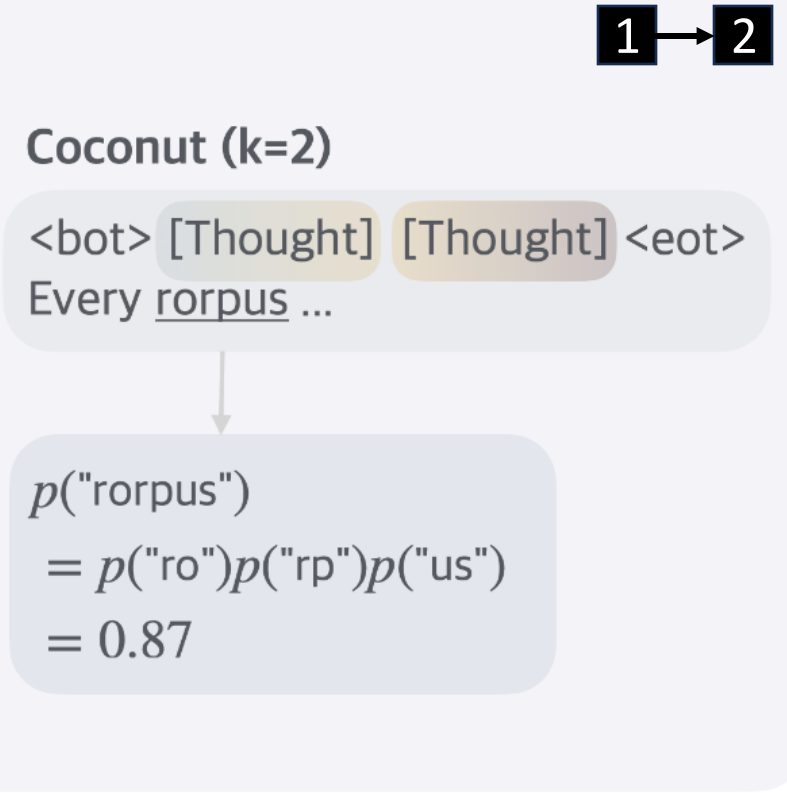
What's inside?

Promising node → dead-end

Interestingly, it encodes all possible paths!



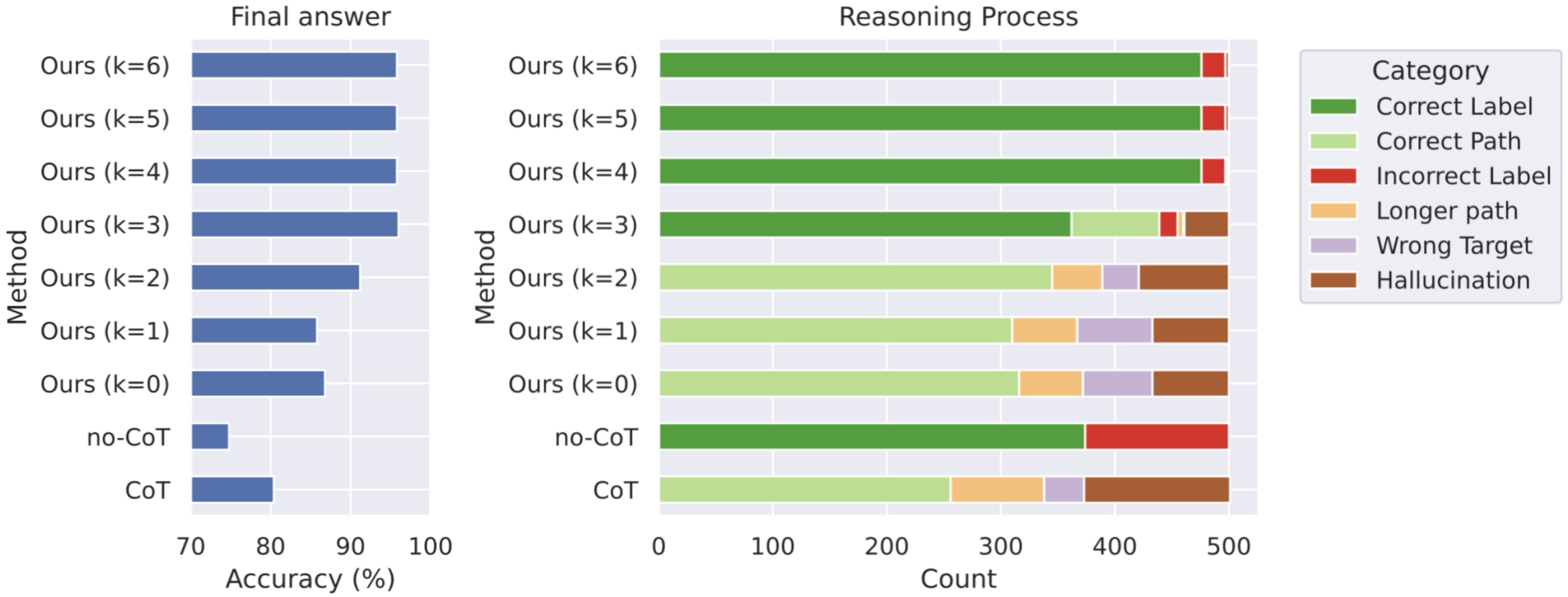




Performance in ProsQA

Dataset	Training	Validation	Test
GSM8k	385,620	500	1319
ProntoQA	9,000	200	800
ProsQA	17,886	300	500

# Nodes	# Edges	Len. of Shortest Path	# Shortest Paths
23.0	36.0	3.8	1.6



CoConut

Method	GSM8k		ProntoQA		ProsQA	
	Acc. (%)	# Tokens	Acc. (%)	# Tokens	Acc. (%)	# Tokens
CoT	42.9 ±0.2	25.0	98.8 ±0.8	92.5	77.5 ±1.9	49.4
No-CoT	16.5 ±0.5	2.2	93.8 ±0.7	3.0	76.7 ±1.0	8.2
iCoT	30.0*	2.2	99.8 ±0.3	3.0	98.2 ±0.3	8.2
Pause Token	16.4 ±1.8	2.2	77.7 ±21.0	3.0	75.9 ±0.7	8.2
COCONUT (Ours)	34.1 ±1.5	8.2	99.8 ±0.2	9.0	97.0 ±0.3	14.2
- *w/o curriculum*	14.4 ±0.8	8.2	52.4 ±0.4	9.0	76.1 ±0.2	14.2
- *w/o thought*	21.6 ±0.5	2.3	99.9 ±0.1	3.0	95.5 ±1.1	8.2
- *pause as thought*	24.1 ±0.7	2.2	100.0 ±0.1	3.0	96.6 ±0.8	8.2

Better performance than No-CoT

Shorter thinking process than CoT

CoConut

Method	GSM8k		ProntoQA		ProsQA	
	Acc. (%)	# Tokens	Acc. (%)	# Tokens	Acc. (%)	# Tokens
CoT	42.9 ±0.2	25.0	98.8 ±0.8	92.5	77.5 ±1.9	49.4
No-CoT	16.5 ±0.5	2.2	93.8 ±0.7	3.0	76.7 ±1.0	8.2
iCoT	30.0*	2.2	99.8 ±0.3	3.0	98.2 ±0.3	8.2
Pause Token	16.4 ±1.8	2.2	77.7 ±21.0	3.0	75.9 ±0.7	8.2
COCONUT (Ours)	34.1 ±1.5	8.2	99.8 ±0.2	9.0	97.0 ±0.3	14.2
- *w/o curriculum*	14.4 ±0.8	8.2	52.4 ±0.4	9.0	76.1 ±0.2	14.2
- *w/o thought*	21.6 ±0.5	2.3	99.9 ±0.1	3.0	95.5 ±1.1	8.2
- *pause as thought*	24.1 ±0.7	2.2				

Better performance than No-CoT
Shorter thinking process than CoT

Cons

1. Latent tokens are not interpretable

2. Only tested on GSM8k

Reasoning Smartly: **Modular Addition**

$$a + b = c \bmod d$$

Does neural network have an *implicit table* to do retrieval?

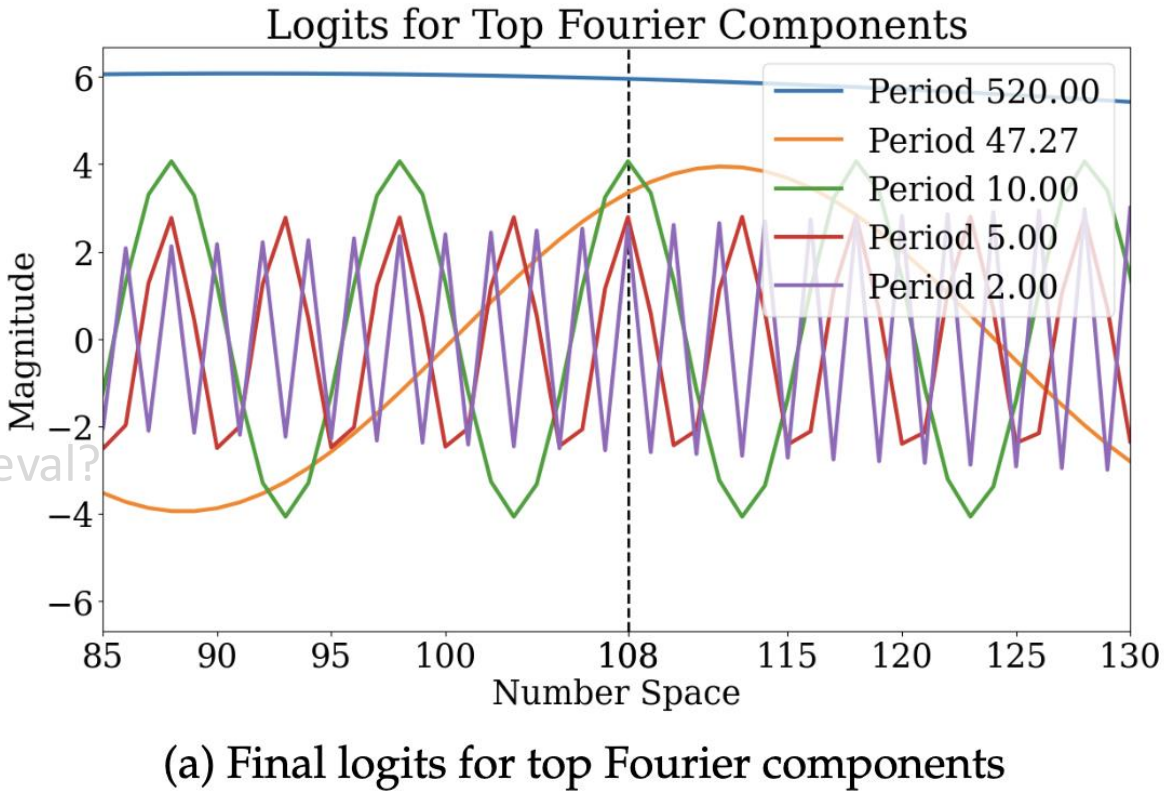
Reasoning Smartly: **Modular Addition**

$$a + b = c \bmod d$$

Does neural network have an *implicit table* to do retrieval?

Learned representation = Fourier basis 😮

Why? 🤔

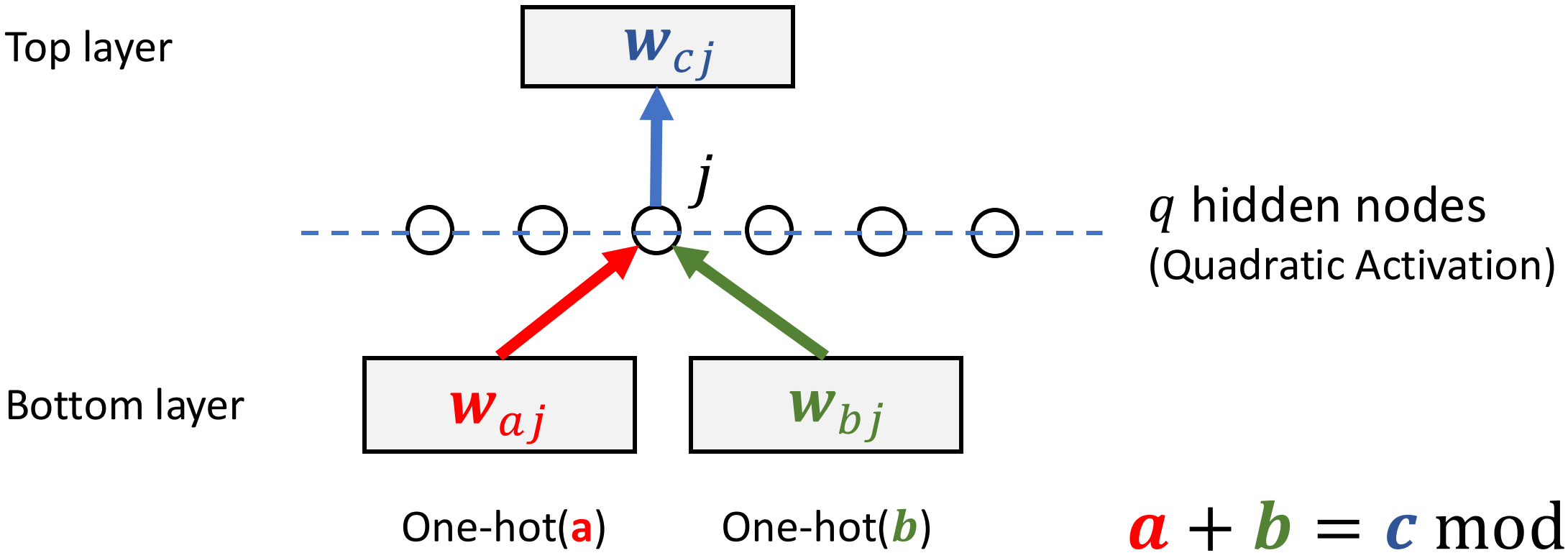


(a) Final logits for top Fourier components

[T. Zhou et al, *Pre-trained Large Language Models Use Fourier Features to Compute Addition*, NeurIPS'24]
[S. Kantamneni, *Language Models Use Trigonometry to Do Addition*, arXiv'25]

Minimal Problem Setup

MSE Loss: $Min \ \|\text{Output} - \text{one-hot}(\boldsymbol{c})\|_2$

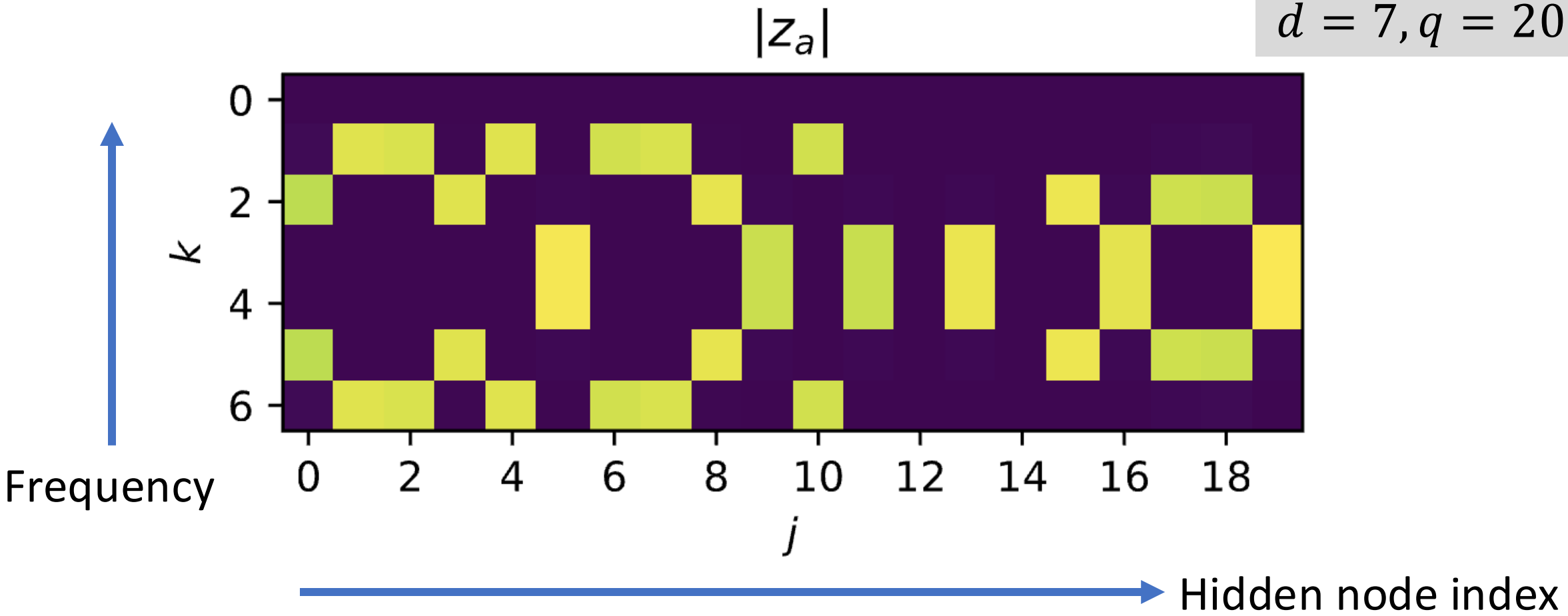


$$\boldsymbol{a} + \boldsymbol{b} = \boldsymbol{c} \bmod d$$

[Y. Tian, *Composing Global Optimizers to Reasoning Tasks via Algebraic Objects in Neural Nets*, arXiv'24]

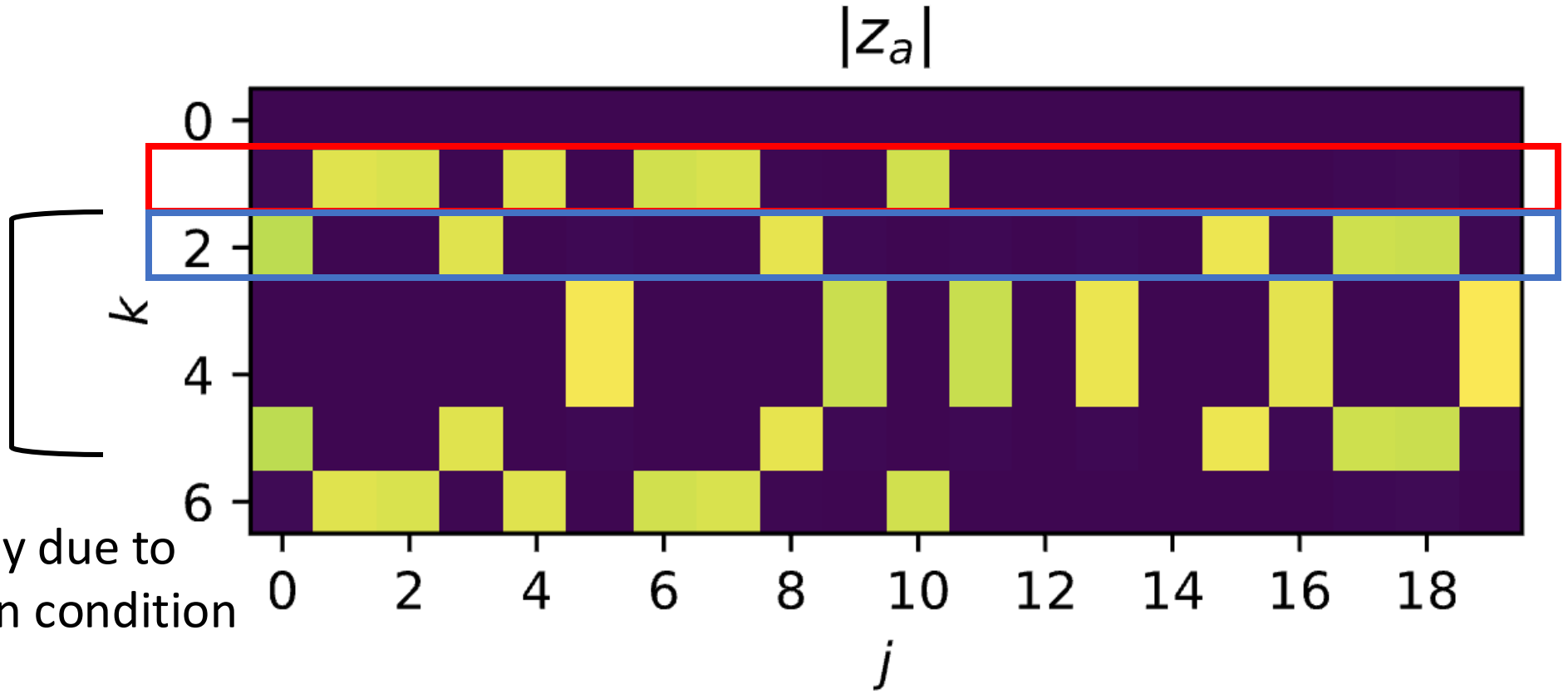
What a Gradient Descent Solution look like?

$d = 7, q = 20$



[Y. Tian, *Composing Global Optimizers to Reasoning Tasks via Algebraic Objects in Neural Nets*, arXiv'24]

What a Gradient Descent Solution look like?

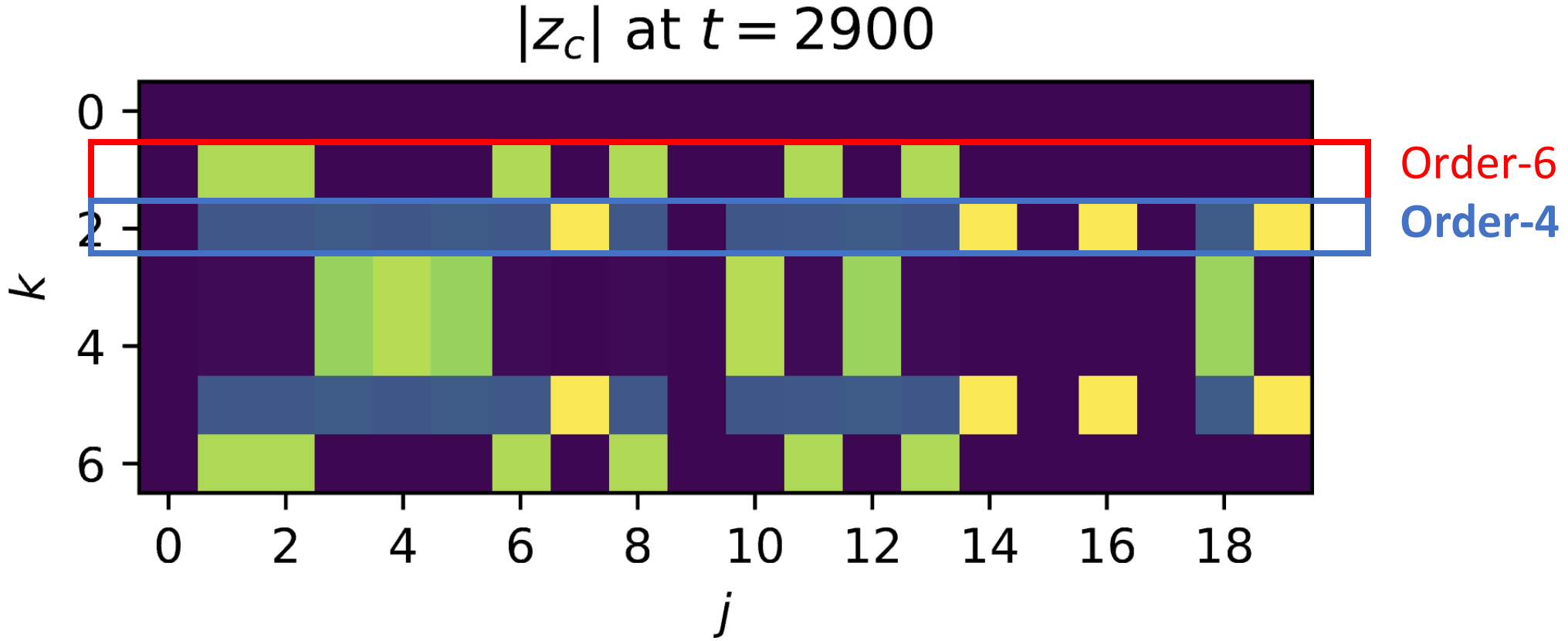


Order-6 solutions

Symmetry due to Hermitian condition

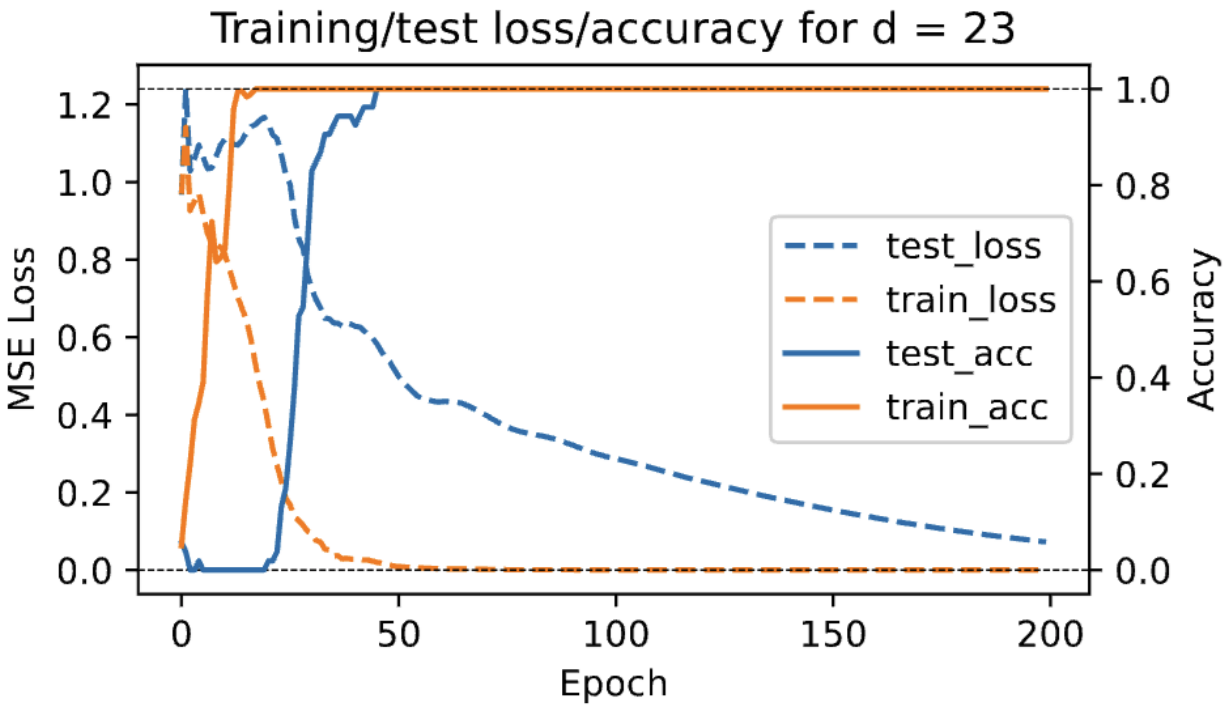
[Y. Tian, *Composing Global Optimizers to Reasoning Tasks via Algebraic Objects in Neural Nets*, arXiv'24]

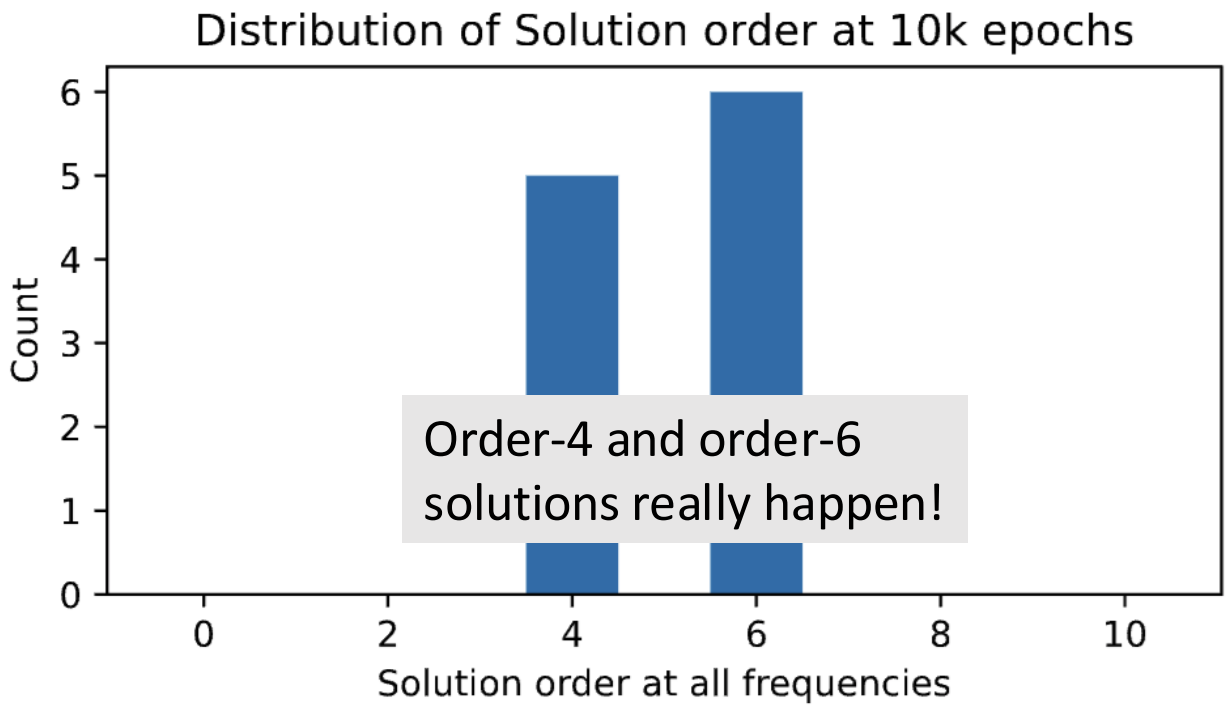
What a Gradient Descent Solution look like?



[Y. Tian, *Composing Global Optimizers to Reasoning Tasks via Algebraic Objects in Neural Nets*, arXiv'24]

More Statistics on Gradient Descent Solutions

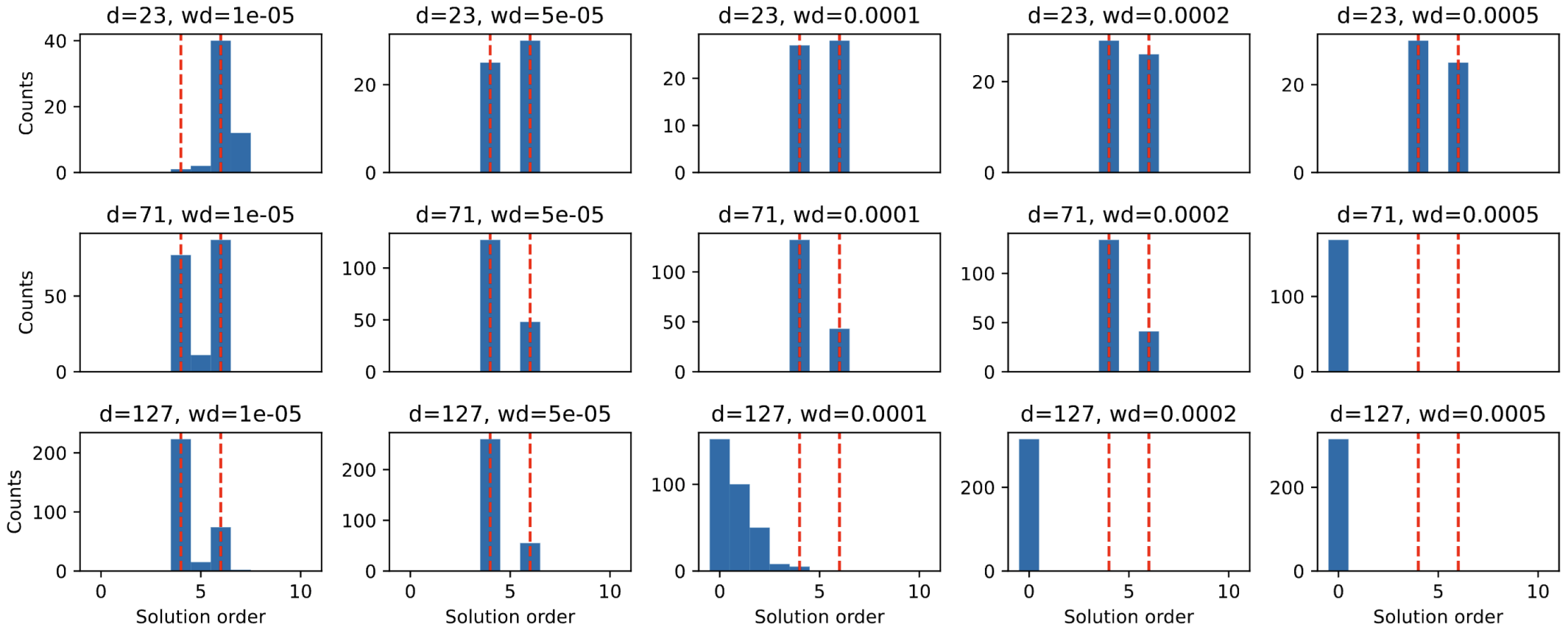




[Y. Tian, *Composing Global Optimizers to Reasoning Tasks via Algebraic Objects in Neural Nets*, arXiv'24]

Effect of Weight Decay

Stronger weight decay



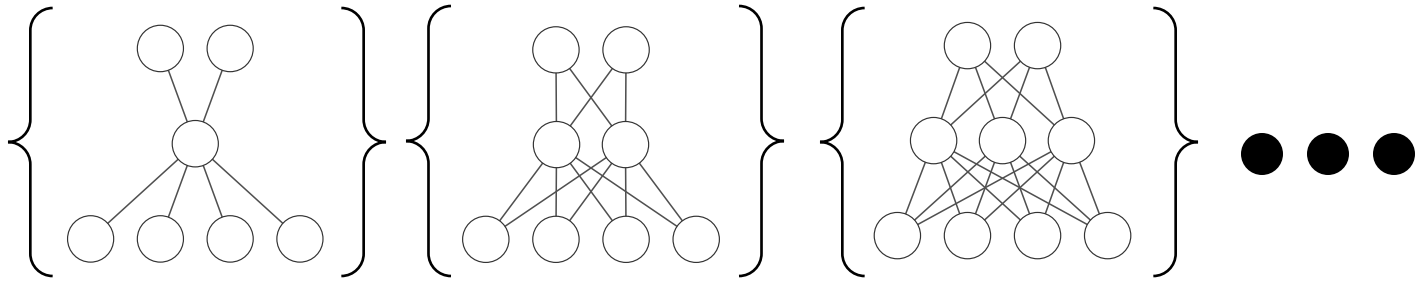
[Y. Tian, *Composing Global Optimizers to Reasoning Tasks via Algebraic Objects in Neural Nets*, arXiv'24]

Why? 

How to Optimize?

The objective is highly nonlinear !!

However, nice *algebraic structures* exist!



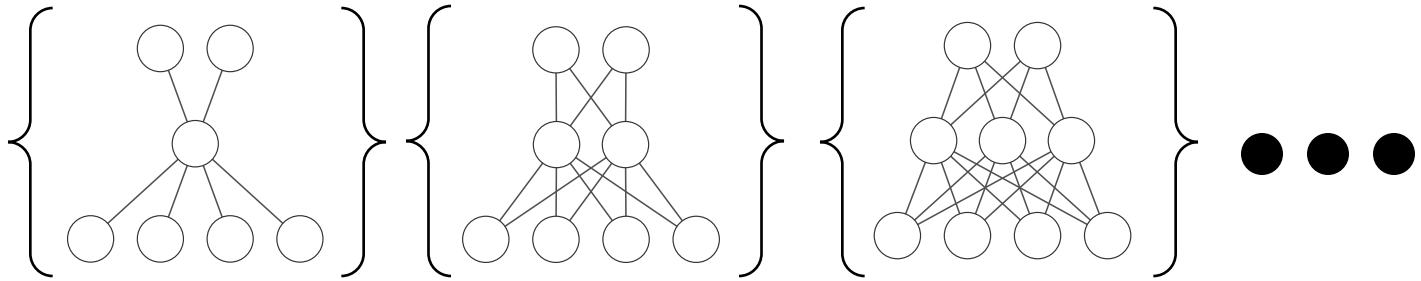
$\mathcal{Z}_1$ $\mathcal{Z}_2$ $\mathcal{Z}_3$

$\mathcal{Z} = \bigcup_{q \geq 0} \mathcal{Z}_q$: All 2-layer networks with different number of hidden nodes

How to Optimize?

The objective is highly nonlinear !!

However, nice *algebraic structures* exist!



$\mathcal{Z} = \bigcup_{q \geq 0} \mathcal{Z}_q$: All 2-layer networks with different number of hidden nodes

Ring addition $+$: Concatenate hidden nodes

Ring multiplication $*$: Kronecker production along the hidden dimensions

$\langle \mathcal{Z}, +, * \rangle$ is a ***semi-ring***

Composing Global Optimizers from Partial Ones

Partial solution #1

$\boldsymbol{z}_{\text{syn}}^{(k)} \in R_{\text{c}} \cap R_{\text{n}}$ but $\boldsymbol{z}_{\text{syn}}^{(k)} \notin R_{*}$

Partial solution #2

$\boldsymbol{z}_{\nu}^{(k)} \in R_{*}$

[Y. Tian, *Composing Global Optimizers to Reasoning Tasks via Algebraic Objects in Neural Nets*, arXiv'24]

Composing Global Optimizers from Partial Ones

Compositing solutions using ***ring multiplication*** $*$

Partial solution #1

$\boldsymbol{z}_{\text{syn}}^{(k)} \in R_{\text{c}} \cap R_{\text{n}}$ but $\boldsymbol{z}_{\text{syn}}^{(k)} \notin R_*$

Partial solution #2

$\boldsymbol{z}_{\nu}^{(k)} \in R_*$

Better solution

$\boldsymbol{z}_{\text{syn}}^{(k)} * \boldsymbol{z}_{\nu}^{(k)} \in R_{\text{c}} \cap R_{\text{n}} \cap R_*$

[Y. Tian, *Composing Global Optimizers to Reasoning Tasks via Algebraic Objects in Neural Nets*, arXiv'24]

Composing Global Optimizers from Partial Ones

Compositing solutions using ***ring multiplication*** $*$

Compositing solutions using ***ring addition*** $+$

Partial solution #1

$\boldsymbol{z}_{\text{syn}}^{(k)} \in R_{\text{c}} \cap R_{\text{n}}$ but $\boldsymbol{z}_{\text{syn}}^{(k)} \notin R_*$

Partial solution #2

$\boldsymbol{z}_{\nu}^{(k)} \in R_*$

Better solution

$\boldsymbol{z}_{\text{syn}}^{(k)} * \boldsymbol{z}_{\nu}^{(k)} \in R_{\text{c}} \cap R_{\text{n}} \cap R_*$

Global Optimizer to MSE loss $\ell(z)$!

$$\boldsymbol{z}_{F6} = \frac{\mathbf{1}}{\sqrt[3]{6}} \sum_k \boldsymbol{z}_{\text{syn}}^{(k)} * \boldsymbol{z}_{\nu}^{(k)}$$

[Y. Tian, *Composing Global Optimizers to Reasoning Tasks via Algebraic Objects in Neural Nets*, arXiv'24]

Exemplar constructed global optimizers

Order-6 $\boldsymbol{z}_{F6}$ (2*3)

$$\boldsymbol{z}_{F6} = \frac{1}{\sqrt[3]{6}} \sum_{k=1}^{(d-1)/2} \boldsymbol{z}_{\mathrm{syn}}^{(k)} * \boldsymbol{z}_{\nu}^{(k)} * \boldsymbol{y}_k$$

Exemplar constructed global optimizers

Order-6 $\boldsymbol{z}_{F6}$ (2*3)

$$\boldsymbol{z}_{F6} = \frac{1}{\sqrt[3]{6}} \sum_{k=1}^{(d-1)/2} \boldsymbol{z}_{\text{syn}}^{(k)} * \boldsymbol{z}_{\nu}^{(k)} * \boldsymbol{y}_k$$

Order-4 $\boldsymbol{z}_{F4/6}$ (2*2)
(mixed with order-6)

$$\boldsymbol{z}_{F4/6} = \frac{1}{\sqrt[3]{6}} \hat{\boldsymbol{z}}_{F6}^{(k_0)} + \frac{1}{\sqrt[3]{4}} \sum_{k=1, k\neq k_0}^{(d-1)/2} \boldsymbol{z}_{F4}^{(k)}$$

Exemplar constructed global optimizers

Order-6 $\boldsymbol{z}_{F6}$ (2*3)

$$\boldsymbol{z}_{F6} = \frac{1}{\sqrt[3]{6}} \sum_{k=1}^{(d-1)/2} \boldsymbol{z}_{\text{syn}}^{(k)} * \boldsymbol{z}_{\nu}^{(k)} * \boldsymbol{y}_k$$

Order-4 $\boldsymbol{z}_{F4/6}$ (2*2)
(mixed with order-6)

$$\boldsymbol{z}_{F4/6} = \frac{1}{\sqrt[3]{6}} \hat{\boldsymbol{z}}_{F6}^{(k_0)} + \frac{1}{\sqrt[3]{4}} \sum_{k=1, k\neq k_0}^{(d-1)/2} \boldsymbol{z}_{F4}^{(k)}$$

Perfect memorization
(order-d per frequency)

$$\boldsymbol{z}_a = \sum_{j=0}^{d-1} \boldsymbol{u}_a^j, \qquad \boldsymbol{z}_b = \sum_{j=0}^{d-1} \boldsymbol{u}_b^j$$

$$\boldsymbol{z}_M = d^{-2/3} \boldsymbol{z}_a * \boldsymbol{z}_b$$

Gradient Descent solutions matches with construction

d	%not order-4/6	%non-factorable order-4	%non-factorable order-6	error ($\times 10^{-2}$) order-4	error ($\times 10^{-2}$) order-6	solution distribution (%) in factorable ones $z^{(k)}_{\nu=\mathrm{i}} * z^{(k)}_{\xi}$	$z^{(k)}_{\nu=\mathrm{i}} * z^{(k)}_{\mathrm{syn},\alpha\beta}$	$z^{(k)}_{\nu} * z^{(k)}_{\mathrm{syn}}$	others
23	$0.0_{\pm 0.0}$	$0.00_{\pm 0.00}$	$5.71_{\pm 5.71}$	$0.05_{\pm 0.01}$	$4.80_{\pm 0.96}$	$47.07_{\pm 1.88}$	$11.31_{\pm 1.76}$	$39.80_{\pm 2.11}$	$1.82_{\pm 1.82}$
71	$0.0_{\pm 0.0}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.03_{\pm 0.00}$	$5.02_{\pm 0.25}$	$72.57_{\pm 0.70}$	$4.00_{\pm 1.14}$	$21.14_{\pm 2.14}$	$2.29_{\pm 1.07}$
127	$0.0_{\pm 0.0}$	$1.50_{\pm 0.92}$	$0.00_{\pm 0.00}$	$0.26_{\pm 0.14}$	$0.93_{\pm 0.18}$	$82.96_{\pm 0.39}$	$2.25_{\pm 0.64}$	$14.13_{\pm 0.87}$	$0.66_{\pm 0.66}$

$$q = 512, wd = 5 \cdot 10^{-5}$$

Gradient Descent solutions matches with construction

d	%not order-4/6	%non-factorable order-4	%non-factorable order-6	error ($\times 10^{-2}$) order-4	error ($\times 10^{-2}$) order-6	solution distribution (%) in factorable ones $z^{(k)}_{\nu=\mathrm{i}} * z^{(k)}_{\xi}$	$z^{(k)}_{\nu=\mathrm{i}} * z^{(k)}_{\mathrm{syn},\alpha\beta}$	$z^{(k)}_{\nu} * z^{(k)}_{\mathrm{syn}}$	others
23	$0.0_{\pm 0.0}$	$0.00_{\pm 0.00}$	$5.71_{\pm 5.71}$	$0.05_{\pm 0.01}$	$4.80_{\pm 0.96}$	$47.07_{\pm 1.88}$	$11.31_{\pm 1.76}$	$39.80_{\pm 2.11}$	$1.82_{\pm 1.82}$
71	$0.0_{\pm 0.0}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.03_{\pm 0.00}$	$5.02_{\pm 0.25}$	$72.57_{\pm 0.70}$	$4.00_{\pm 1.14}$	$21.14_{\pm 2.14}$	$2.29_{\pm 1.07}$
127	$0.0_{\pm 0.0}$	$1.50_{\pm 0.92}$	$0.00_{\pm 0.00}$	$0.26_{\pm 0.14}$	$0.93_{\pm 0.18}$	$82.96_{\pm 0.39}$	$2.25_{\pm 0.64}$	$14.13_{\pm 0.87}$	$0.66_{\pm 0.66}$

100% of the per-freq solutions are order-4/6

Gradient Descent solutions matches with construction

d	%not order-4/6	%non-factorable order-4	%non-factorable order-6	error ($\times 10^{-2}$) order-4	error ($\times 10^{-2}$) order-6	solution distribution (%) in factorable ones $z^{(k)}_{\nu=\mathrm{i}} * z^{(k)}_{\xi}$	$z^{(k)}_{\nu=\mathrm{i}} * z^{(k)}_{\mathrm{syn},\alpha\beta}$	$z^{(k)}_{\nu} * z^{(k)}_{\mathrm{syn}}$	others
23	$0.0_{\pm 0.0}$	$0.00_{\pm 0.00}$	$5.71_{\pm 5.71}$	$0.05_{\pm 0.01}$	$4.80_{\pm 0.96}$	$47.07_{\pm 1.88}$	$11.31_{\pm 1.76}$	$39.80_{\pm 2.11}$	$1.82_{\pm 1.82}$
71	$0.0_{\pm 0.0}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.03_{\pm 0.00}$	$5.02_{\pm 0.25}$	$72.57_{\pm 0.70}$	$4.00_{\pm 1.14}$	$21.14_{\pm 2.14}$	$2.29_{\pm 1.07}$
127	$0.0_{\pm 0.0}$	$1.50_{\pm 0.92}$	$0.00_{\pm 0.00}$	$0.26_{\pm 0.14}$	$0.93_{\pm 0.18}$	$82.96_{\pm 0.39}$	$2.25_{\pm 0.64}$	$14.13_{\pm 0.87}$	$0.66_{\pm 0.66}$

95% of the solutions are factorizable into “2*3” or “2*2”

Gradient Descent solutions matches with construction

d	%not order-4/6	%non-factorable order-4	%non-factorable order-6	error ($\times 10^{-2}$) order-4	error ($\times 10^{-2}$) order-6	solution distribution (%) in factorable ones $\boldsymbol{z}^{(k)}_{\nu=\mathrm{i}} * \boldsymbol{z}^{(k)}_{\xi}$	solution distribution (%) in factorable ones $\boldsymbol{z}^{(k)}_{\nu=\mathrm{i}} * \boldsymbol{z}^{(k)}_{\mathrm{syn},\alpha\beta}$	solution distribution (%) in factorable ones $\boldsymbol{z}^{(k)}_{\nu} * \boldsymbol{z}^{(k)}_{\mathrm{syn}}$	solution distribution (%) in factorable ones others
23	$0.0_{\pm 0.0}$	$0.00_{\pm 0.00}$	$5.71_{\pm 5.71}$	$0.05_{\pm 0.01}$	$4.80_{\pm 0.96}$	$47.07_{\pm 1.88}$	$11.31_{\pm 1.76}$	$39.80_{\pm 2.11}$	$1.82_{\pm 1.82}$
71	$0.0_{\pm 0.0}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.03_{\pm 0.00}$	$5.02_{\pm 0.25}$	$72.57_{\pm 0.70}$	$4.00_{\pm 1.14}$	$21.14_{\pm 2.14}$	$2.29_{\pm 1.07}$
127	$0.0_{\pm 0.0}$	$1.50_{\pm 0.92}$	$0.00_{\pm 0.00}$	$0.26_{\pm 0.14}$	$0.93_{\pm 0.18}$	$82.96_{\pm 0.39}$	$2.25_{\pm 0.64}$	$14.13_{\pm 0.87}$	$0.66_{\pm 0.66}$

Factorization error is very small

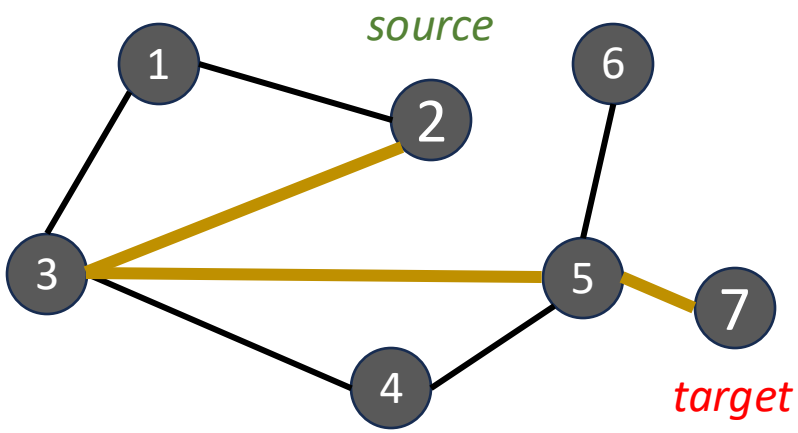
Gradient Descent solutions matches with construction

d	%not order-4/6	%non-factorable order-4	%non-factorable order-6	error ($\times 10^{-2}$) order-4	error ($\times 10^{-2}$) order-6	solution distribution (%) in factorable ones $\boldsymbol{z}^{(k)}_{\nu=\mathrm{i}} * \boldsymbol{z}^{(k)}_{\xi}$	solution distribution (%) in factorable ones $\boldsymbol{z}^{(k)}_{\nu=\mathrm{i}} * \boldsymbol{z}^{(k)}_{\mathrm{syn},\alpha\beta}$	solution distribution (%) in factorable ones $\boldsymbol{z}^{(k)}_{\nu} * \boldsymbol{z}^{(k)}_{\mathrm{syn}}$	solution distribution (%) in factorable ones others
23	$0.0_{\pm 0.0}$	$0.00_{\pm 0.00}$	$5.71_{\pm 5.71}$	$0.05_{\pm 0.01}$	$4.80_{\pm 0.96}$	$47.07_{\pm 1.88}$	$11.31_{\pm 1.76}$	$39.80_{\pm 2.11}$	$1.82_{\pm 1.82}$
71	$0.0_{\pm 0.0}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.03_{\pm 0.00}$	$5.02_{\pm 0.25}$	$72.57_{\pm 0.70}$	$4.00_{\pm 1.14}$	$21.14_{\pm 2.14}$	$2.29_{\pm 1.07}$
127	$0.0_{\pm 0.0}$	$1.50_{\pm 0.92}$	$0.00_{\pm 0.00}$	$0.26_{\pm 0.14}$	$0.93_{\pm 0.18}$	$82.96_{\pm 0.39}$	$2.25_{\pm 0.64}$	$14.13_{\pm 0.87}$	$0.66_{\pm 0.66}$

98% of the solutions can be factorizable into the constructed forms

Shortest Path: Symbolic Emerged from Neural Rep

Task: Learn a 2-layer Transformer for predicting **shortest path** in the graph



<bos> 1 2 <e> … <q> [source] [target] <p> [source] [node 1] [node 2] … [target]

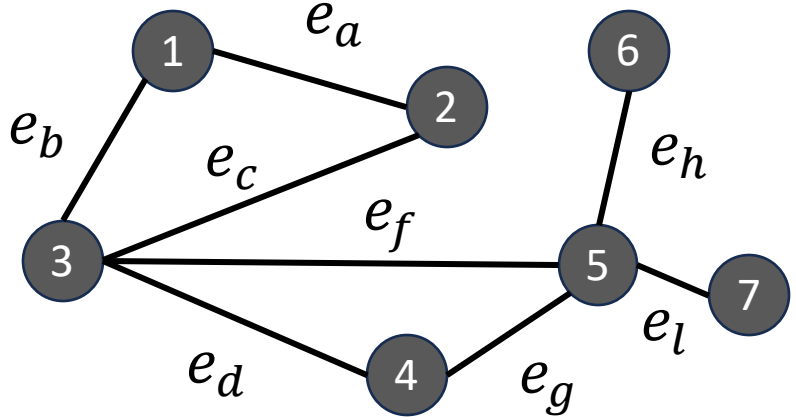
Context

Predicted Shortest path

[A. Cohen et al, *Spectral Journey: How Transformers Predict the Shortest Path,* arXiv'25]

What representations it learns?

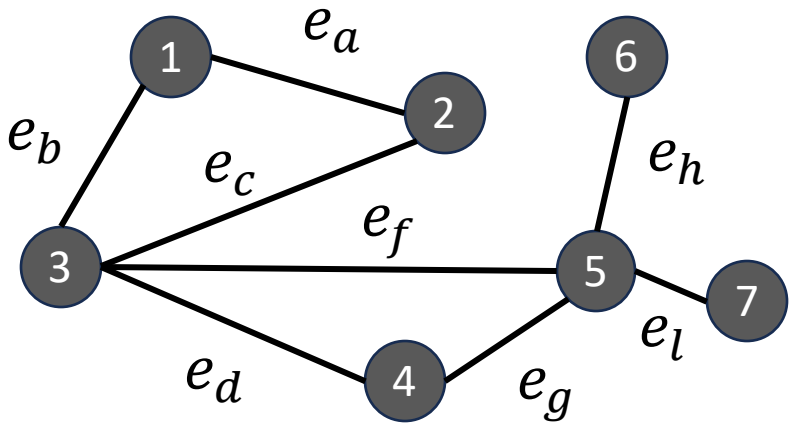
Neural Representation



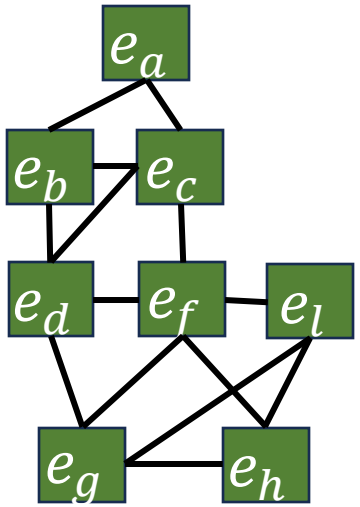
Representation after the
first Transformer layer
(averaged over random edge order)

<bos> 1 2 **<e>** … <q> [source] [target] <p> [source] [node 1] [node 2] … [target]

What representations it learns?



Line graph



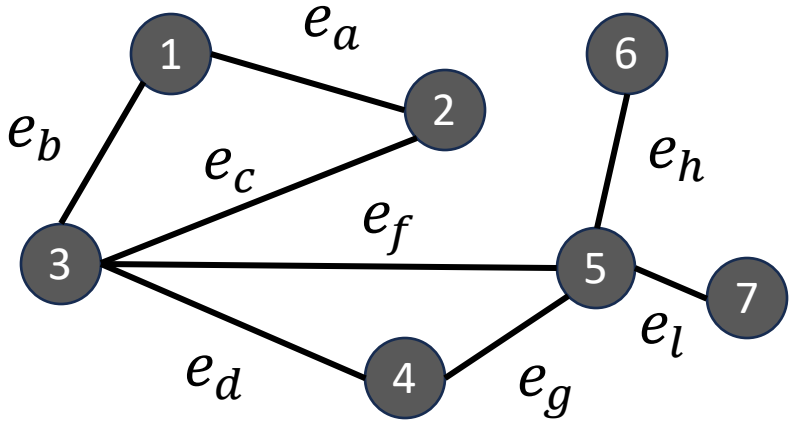
Normalized Graph Laplacian

Symbolic Representation

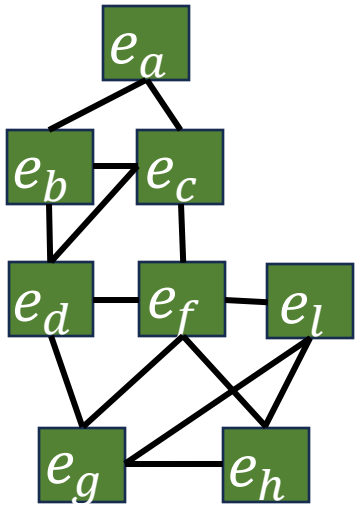
$$L = I - D^{-1/2} A D^{-1/2}$$

Edge Embedding

What representations it learns?



Line graph



Normalized Graph Laplacian

$$L = I - D^{-1/2} A D^{-1/2}$$

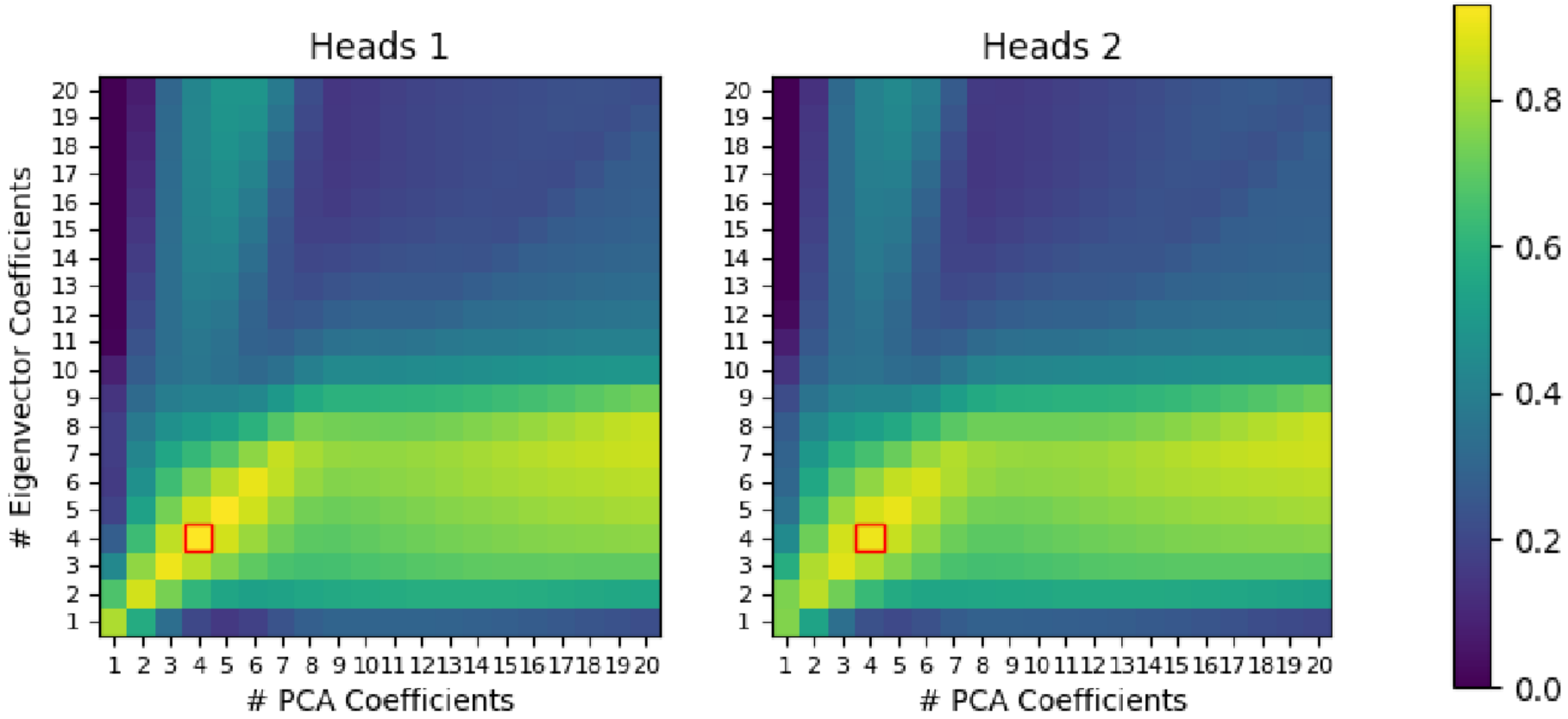
Representation after the first Transformer layer (averaged over random edge order)

Edge Embedding

<bos> 1 2 **<e>** … <q> [source] [target] <p> [source] [node 1] [node 2] … [target]

What representations it learns?

Graph Edge Embedding of various dimensions



Computed edge embedding with trained Transformers

Normalized Correlation > 0.9

Spectral Line Navigator (SLN)

Simple Algorithms of Graph Shortest Path

1. Compute Line Graph $\tilde{G}$ of existing graph G
2. Compute eigenvectors of normalized Laplacian $L(\tilde{G})$
3. $i = source$
4. While $i \neq target$ do

   $distance(j,k;i) := \left\| v_{ij} - v_{k,target} \right\|_2$

   Find $j = \text{argmin}_{j,k}\, distance(j,k;i)$

   Let $i = j$

>99% optimal for small random graph (size < 10)

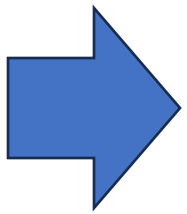
o3-mini-high implementation: https://chatgpt.com/share/67b027f9-fb28-8012-aa64-a1f7479134b7

Possible Implications

Do neural networks end up learning more efficient **symbolic representations** that we don't know?

Does gradient descent lead to a solution that can be reached by **advanced algebraic operations**?

Will gradient descent become **obsolete**, eventually?



Thanks!