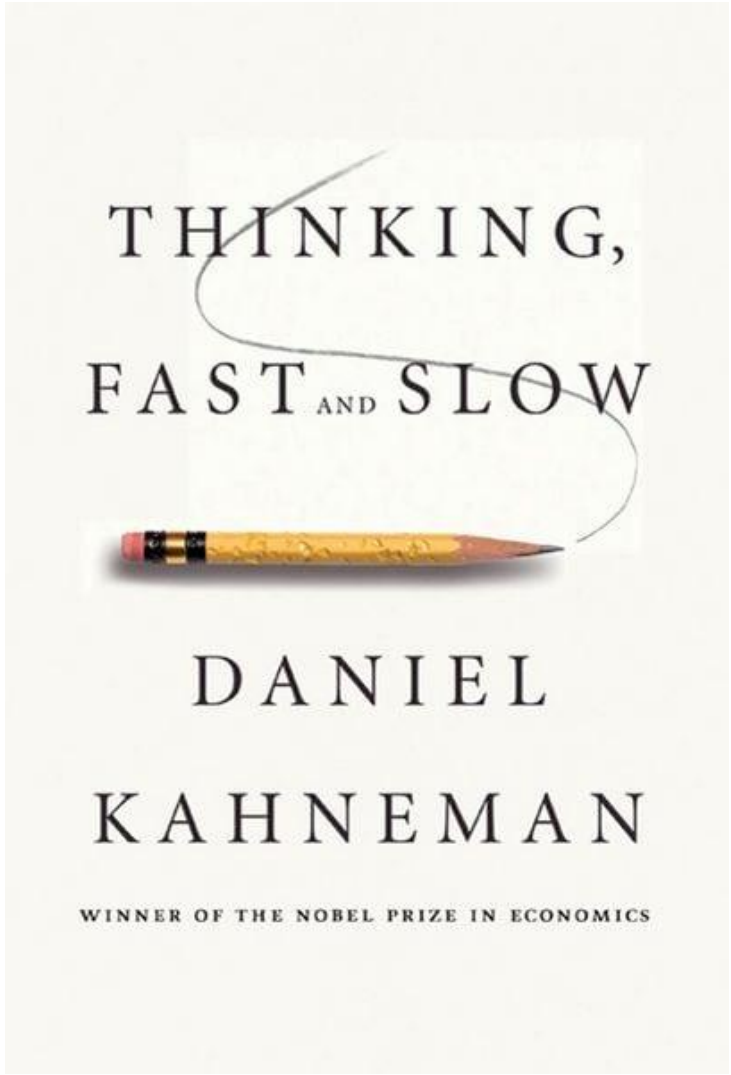
ETH zürich

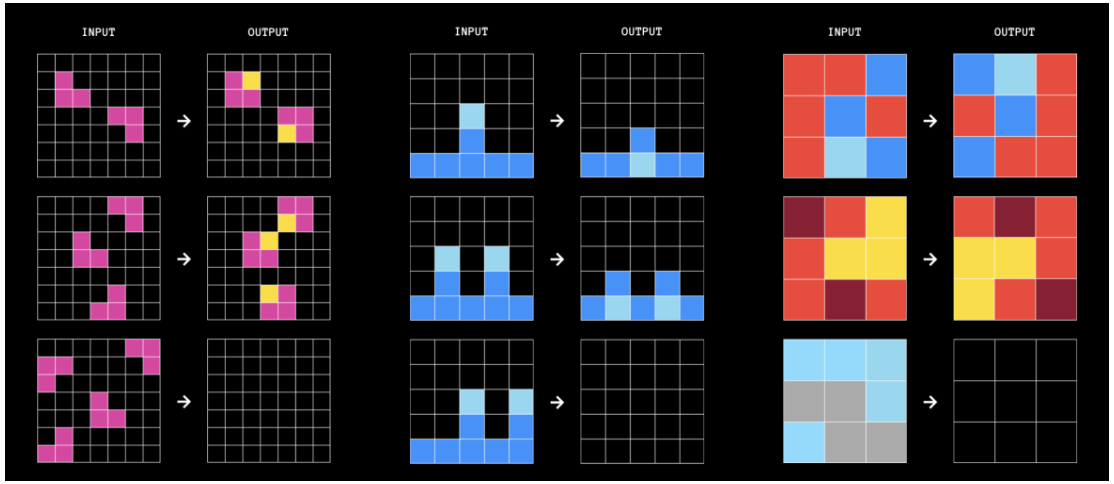
Universal Reasoning Model

Phillip Trummer

05 May 2026


THINKING,
FAST AND SLOW
DANIEL
KAHNEMAN
WINNER OF THE NOBEL PRIZE IN ECONOMICS

ARC-AGI



What does ARC-AGI test?

- Abstract pattern recognition from minimal examples
- Few-shot generalization to novel visual transformations
- Tasks require reasoning, not memorization

Universal Transformers (UT)

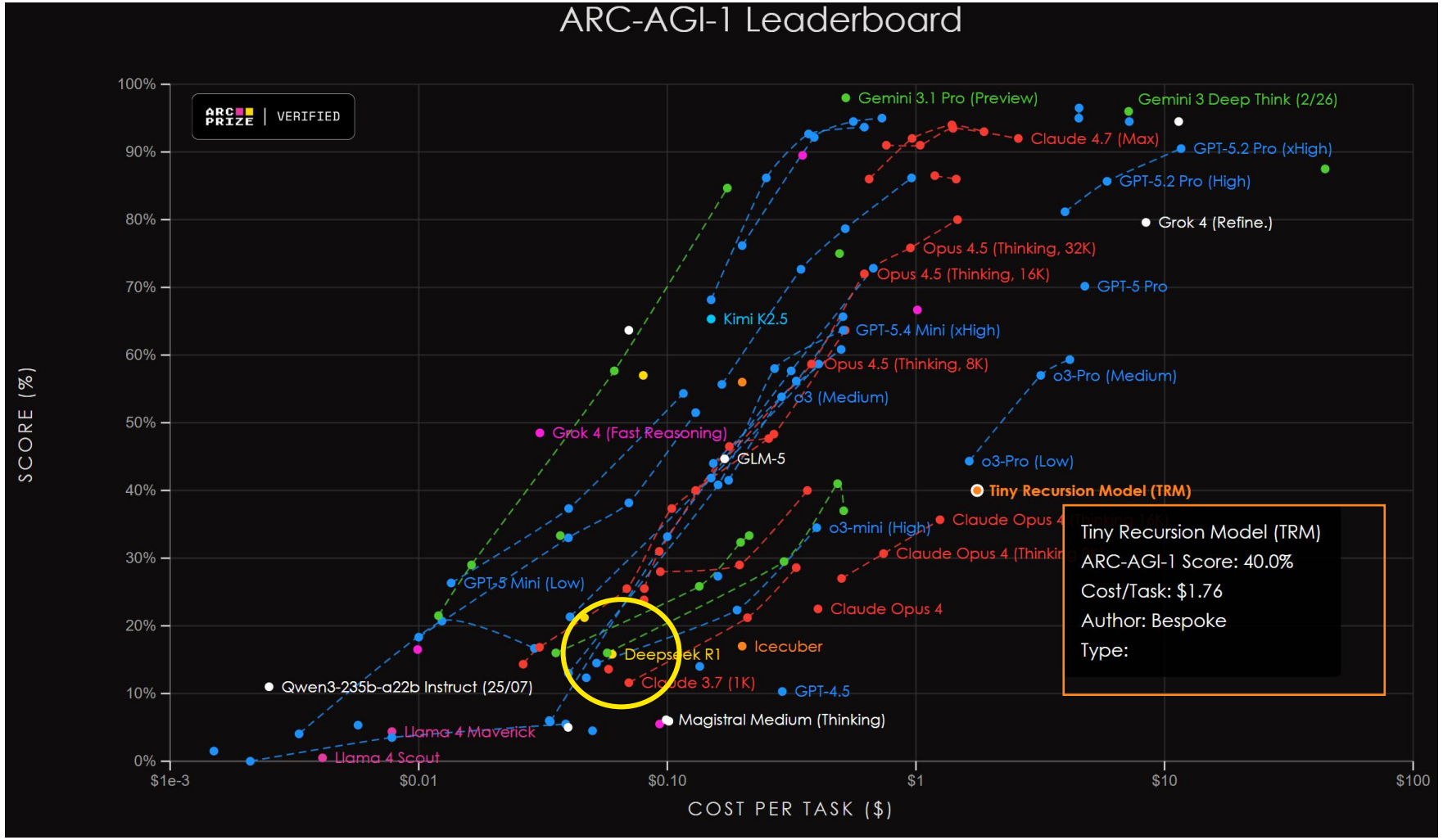
Model	Parameter Size	ARC-AGI 1	ARC-AGI 2
Deepseek R1	~671B	**15.8%	**1.3%
***Standard Transformers** (32 layers, 512 hidden)	~64M	23.75%	-
***Standard Transformers** (4 layers, 512 hidden)	~8M	5.13%	-
***URM** (Universal Reasoning Model)	~15M	53.8%	16.0%
***TRM** (Tiny Recursive Model)	~7M	40.0%	4.6%
***HRM** (Hierarchical Reasoning Model)	~27M	34.4%	5.4%

Table 1: ARC-AGI performance comparison.

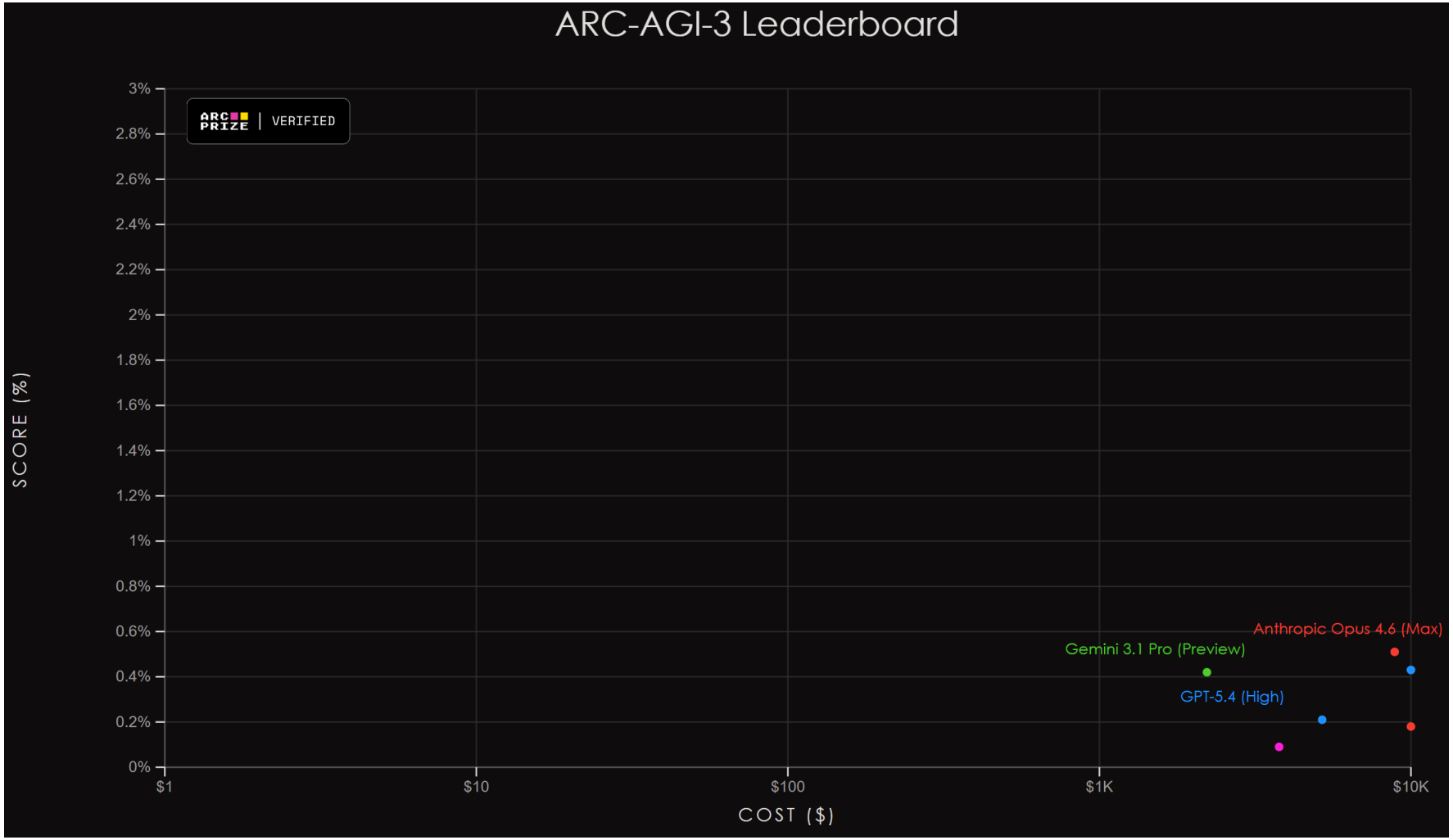
* trained from scratch on task-specific data, no pre-training
** pass@2; other models compared at pass@1

[1] Dehghani, M., et al (2019). *Universal Transformers*. [2] Gao, Z., et al. (2025). *Universal Reasoning Model*. [3] Jolicoeur-Martineau, A. (2025). *Less is More: Recursive Reasoning with Tiny Networks*.

The storyline has shifted…



… has it?



ARC Prize – Leaderboard (https://arcprize.org/leaderboard)

Why is Sudoku so hard for LLMs?

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

QUESTION
53..7….6..195….98….6.8…6…34..8.3..17…2…6.6….28….419..5….8..79

ANSWER
534678912672195348198342567859761423426853791713924856961537284287419635345286179

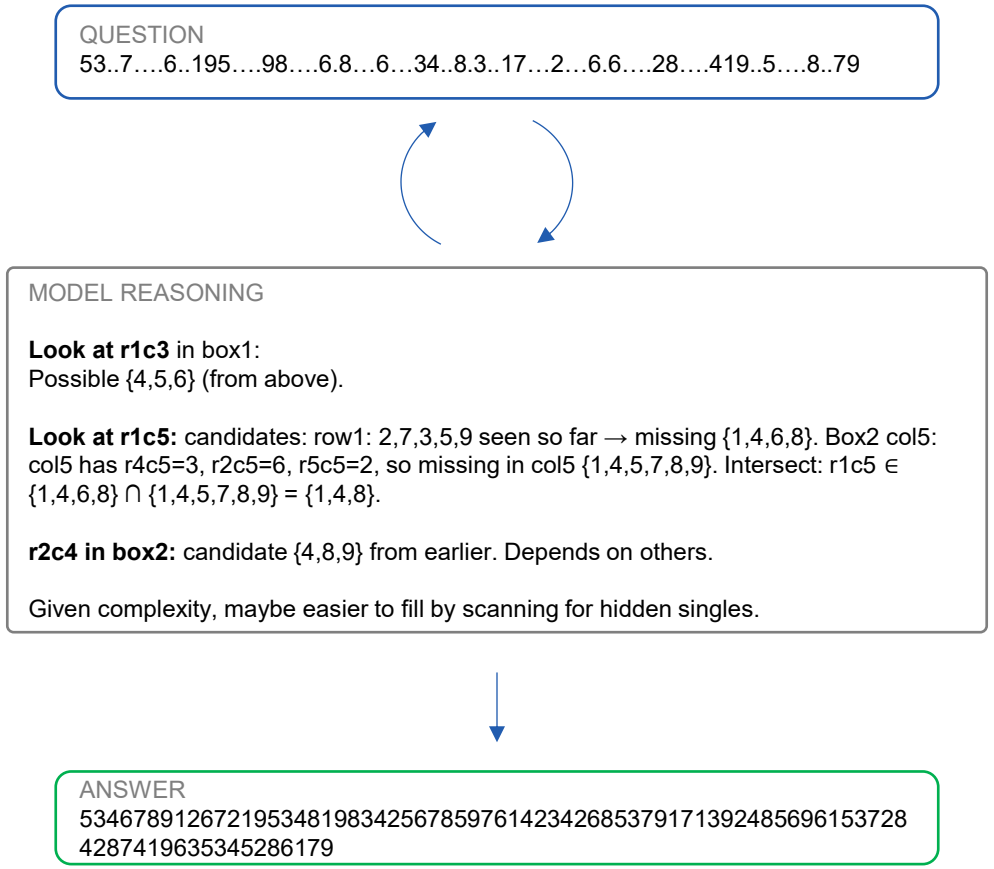
Model	Parameter Size	Sudoku (Extreme)
Deepseek R1	~671B	0%
***URM** (Universal Reasoning Model)	~15M (~0.0022% of Deepseek)	77.6%
***TRM** (Tiny Recursive Model)	~7M (~0.0010%)	66.8%
***HRM** (Hierarchical Reasoning Model)	~27M (~0.0040%)	63.9%

Table 2: Sudoku performance comparison.

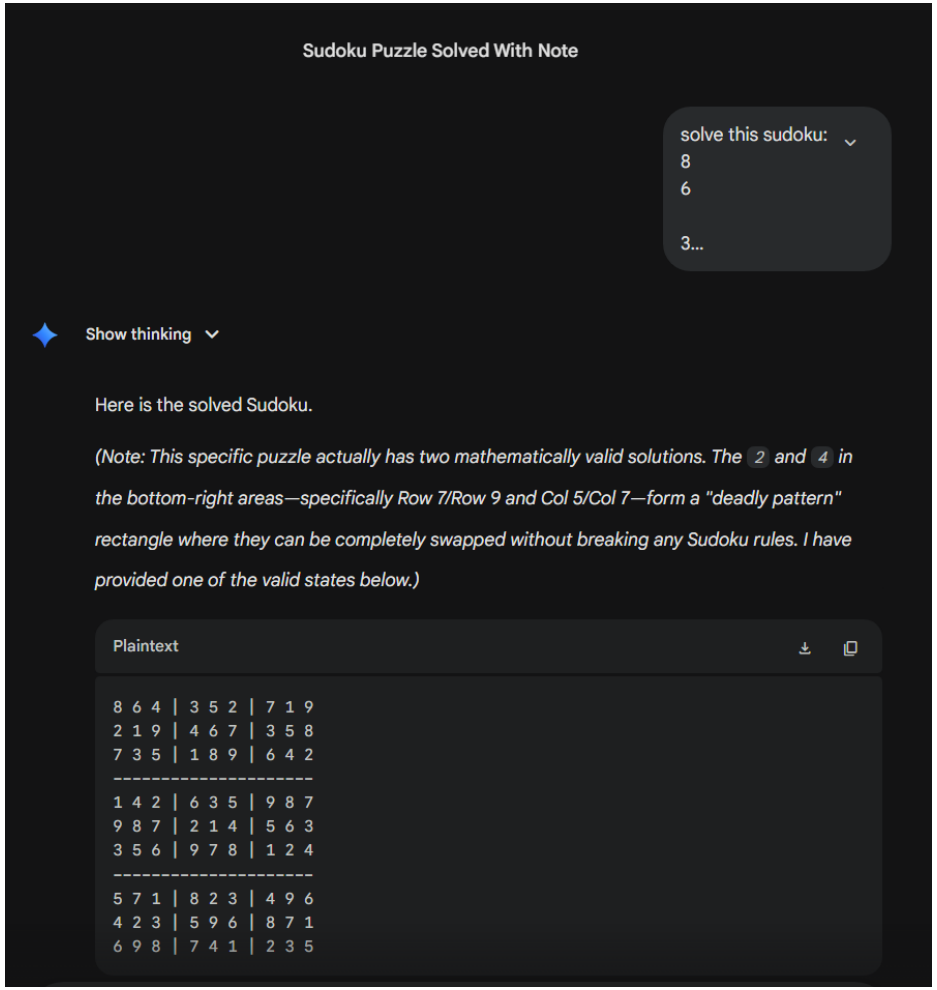
*models trained from scratch on task-specific data, no pre-training

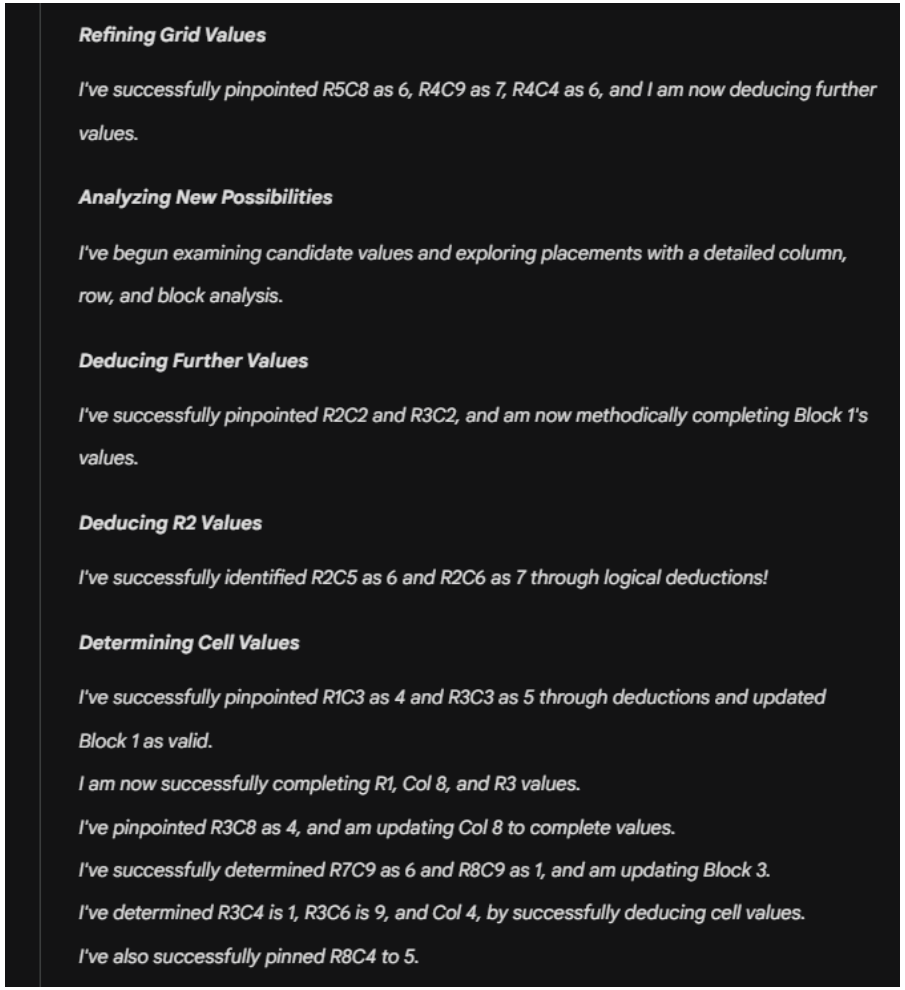
[2] Gao, Z., et al. (2025). *Universal Reasoning Model*. [3] Jolicoeur-Martineau, A. (2025). *Less is More: Recursive Reasoning with Tiny Networks*.

Why is Sudoku so hard for LLMs?



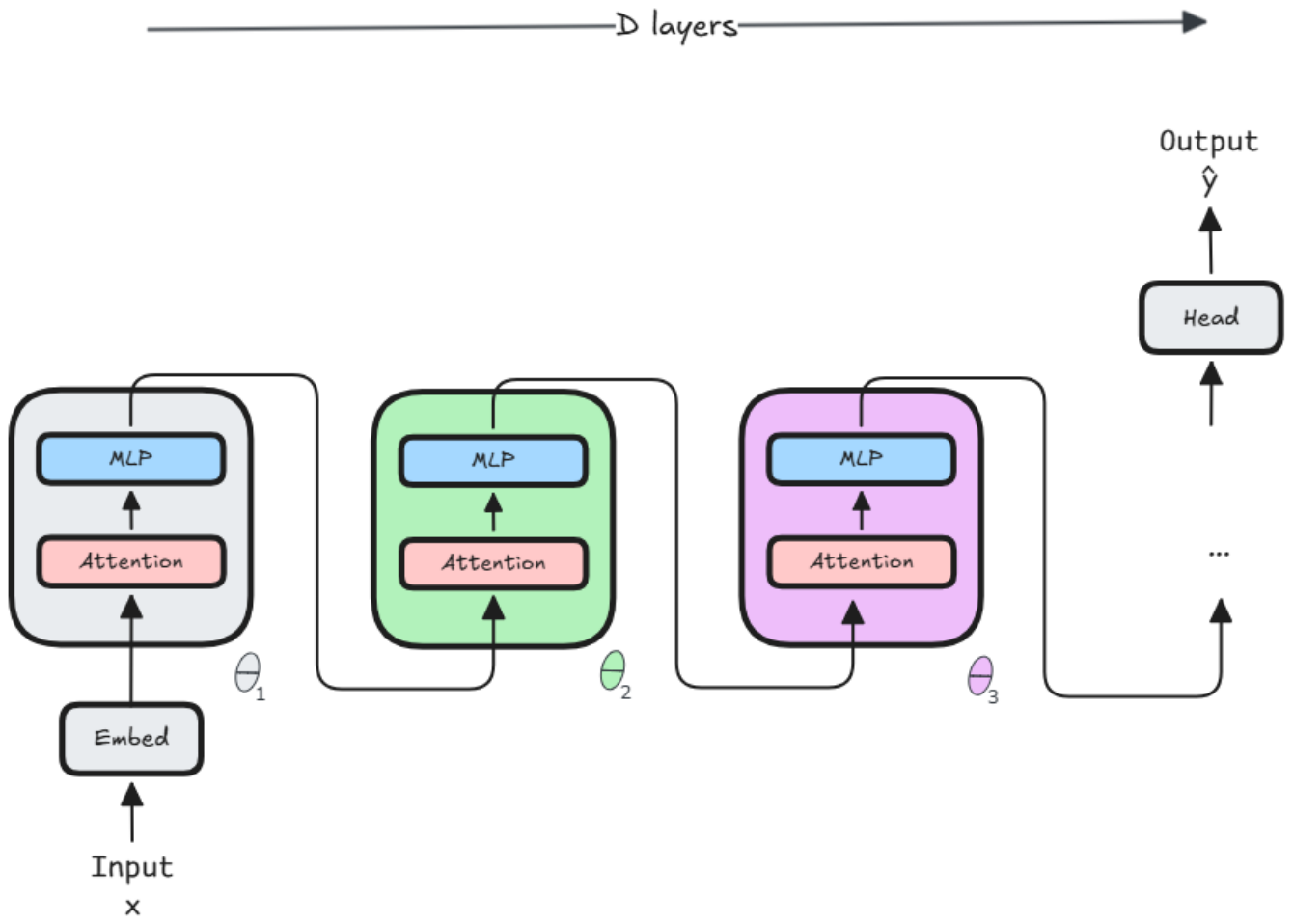
… not for Gemini 3.1 pro



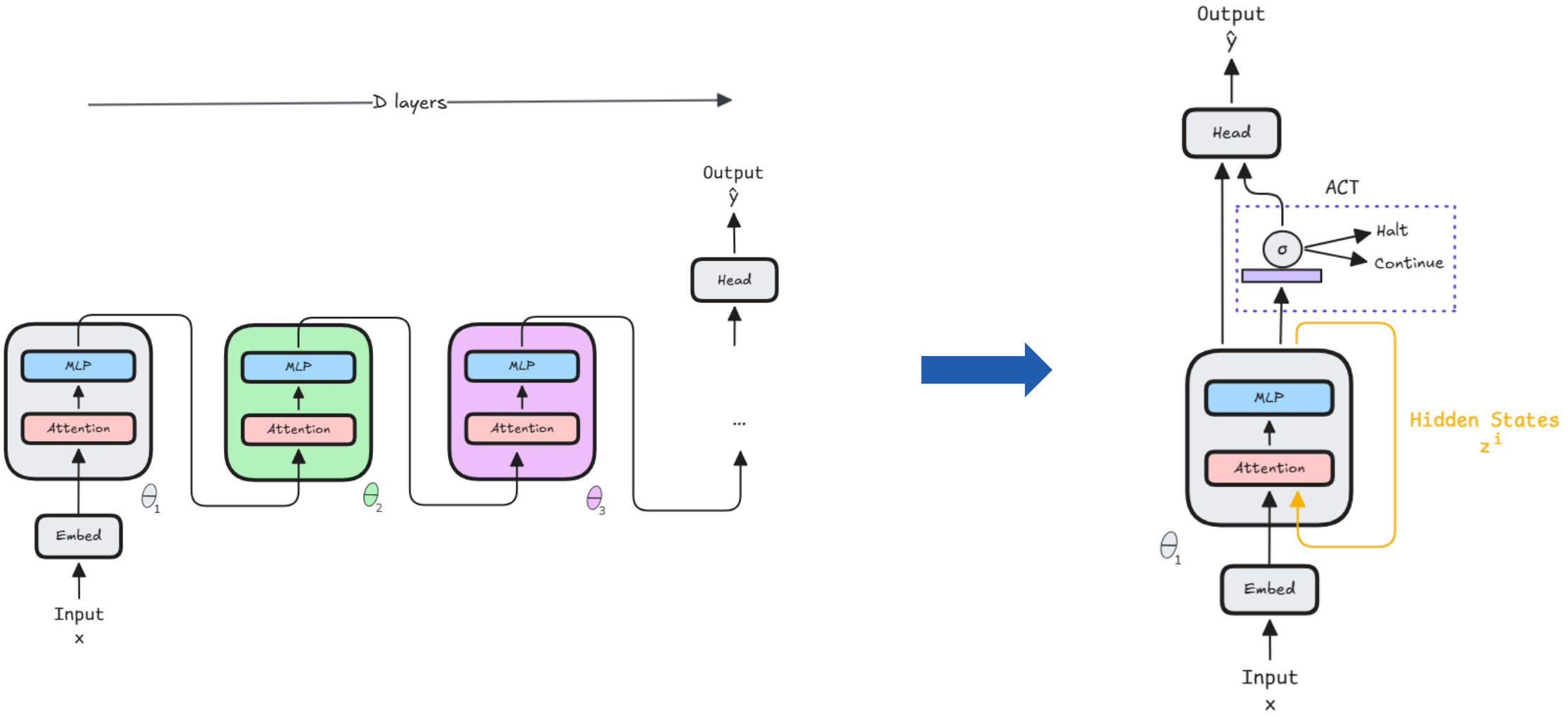


Vanilla Transformers

*simplified



Inducing latent reasoning: **Recurrent Network (shared weights) with Recall**



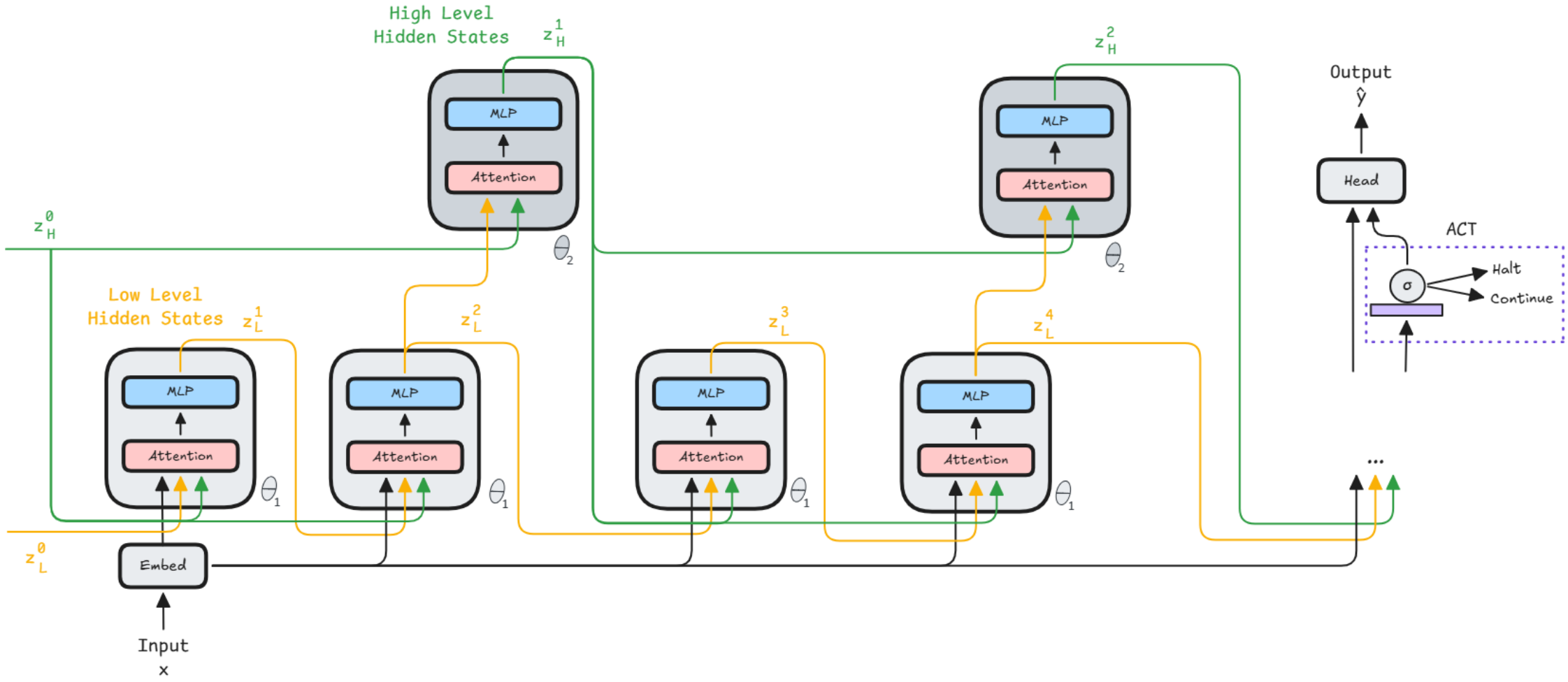
Universal Transformers (UT) – **Why do they work so well?**

1. **Theoretical expressivity.** Can spend variable amount of compute on different inputs.

2. **Parameter efficiency.** D-layers effective depth with 1 layer of parameters.

Hierarchical Reasoning Model (HRM)

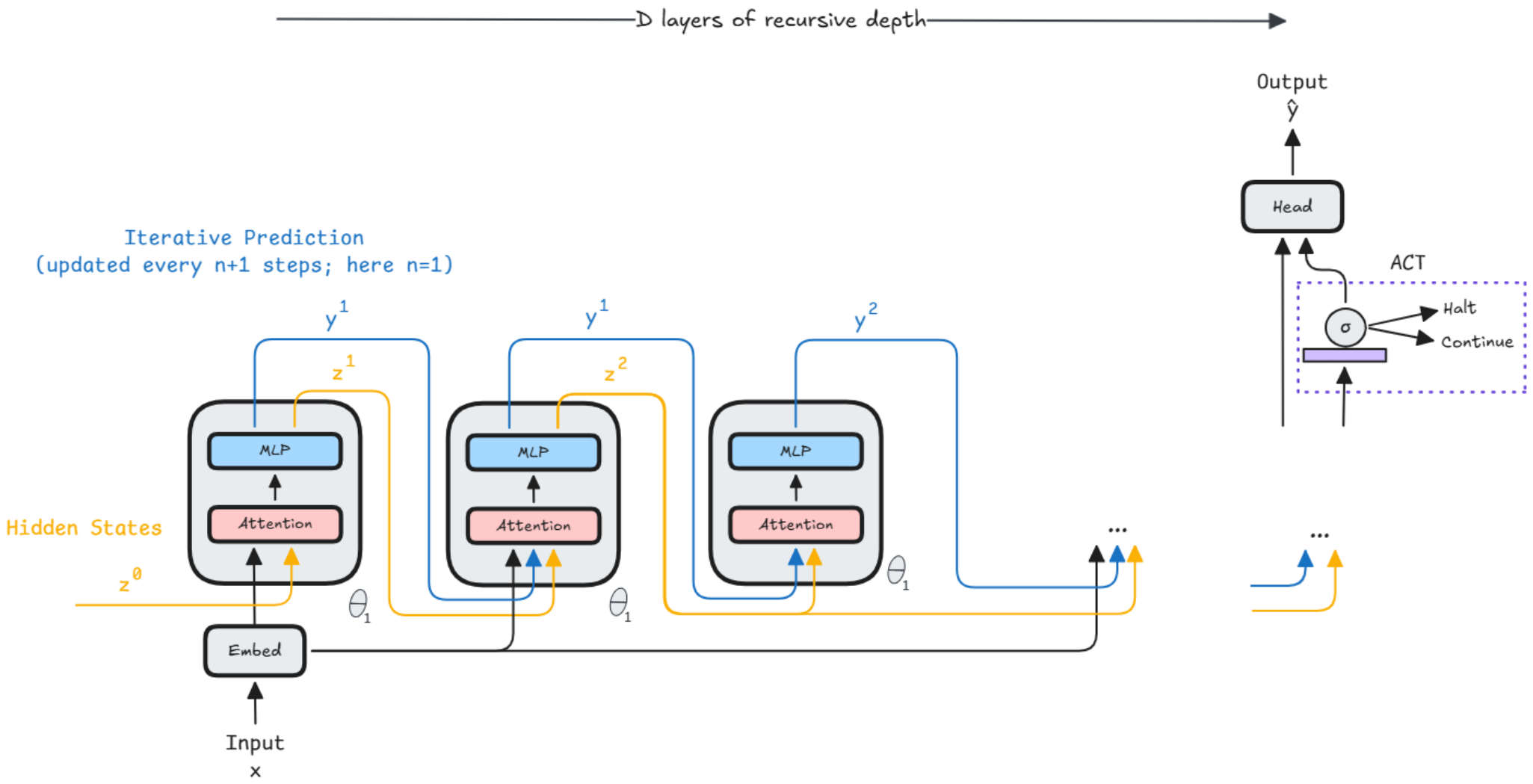
*simplified



[3] Wang, G., et al. (2025). *Hierachical Reasoning Model*.

Tiny Reasoning Model (TRM)

*simplified



[2] Jolicoeur-Martineau, A. (2025). *Less is More: Recursive Reasoning with Tiny Networks*.

Where do the gains come from and can we generalize the architecture?

Systematic ablation analysis: Methodically stripping away components to find what matters

- Removing attention softmax → **catastrophic failure**
- Removing MLP activations → **catastrophic failure**
- Substituting SwiGLU (Swish Gated Linear Unit) for SiLU or ReLU → **catastrophic failure**

$$\mathrm{FFN}_{\mathrm{SwiGLU}}(x) = (\mathrm{SiLU}(xW_1) \otimes xW_2)W_3$$

$$\mathrm{ReLU}: f(x) = \max(0, x)$$

$$\mathrm{SiLU}: f(x) = \frac{x}{1+e^{-x}}$$

- Decreasing recurrent loops → **monotonic decline** in accuracy

[4] Shazeer, N. (2020): *GLU Variants Improve Transformer*

Where do the gains come from and can we generalize the architecture?

The **nonlinearity** of the model (gating + activations like SwiGLU/SiLU).

The **recurrent mechanism** (iterative refinement via weight sharing).

Universal Reasoning Model (URM)

Modification 1: ConvSwiGLU

The problem:

Standard SwiGLU is point-wise → tokens cannot mix information

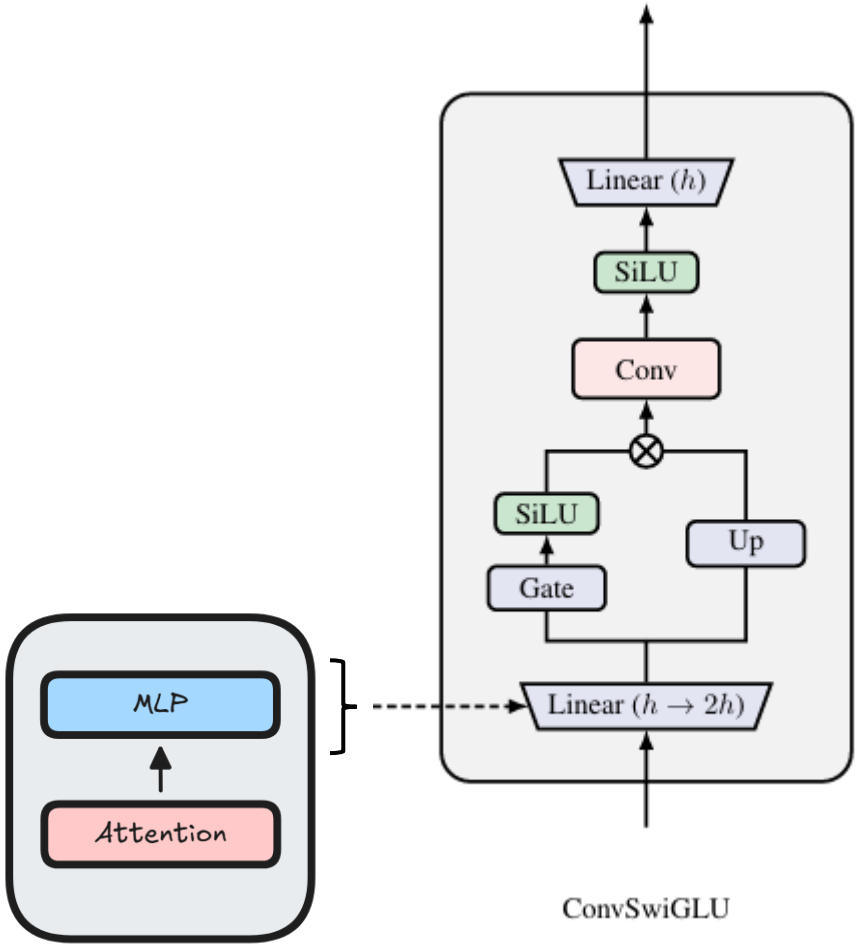
The solution:

- Insert lightweight **depthwise 1D convolution** inside the MLP block

$$FFN_{ConvSwiGLU}(x) = SiLU\big(Conv1D(SiLU(xW_1) \otimes xW_2)\big)W_3$$

Key findings:

- Works in MLP → **more diverse, structured attention patterns**
- Fails in attention → **distorts key-query structure**



Universal Reasoning Model (URM)

Modification 2: Truncated Backpropagation Through Loops (TBPTL)

Truncated Backpropagation Through Loops

Training with many recurrent loops causes **unstable gradients** (the classic RNN problem).

Solution:

- Don't backpropagate through the first few loops
- Run initial iterations in **forward-only mode** to stabilize training

Sweet spot:

- 8 total loops, **2 forward-only** = best performance
- Too much truncation → degraded multi-step refinement
- Too little → noisy, unstable gradients

Where do the gains come from and can we generalize the architecture?

The **nonlinearity** of the model

⟹ **ConvSwiGLU**

The **recurrent mechanism**

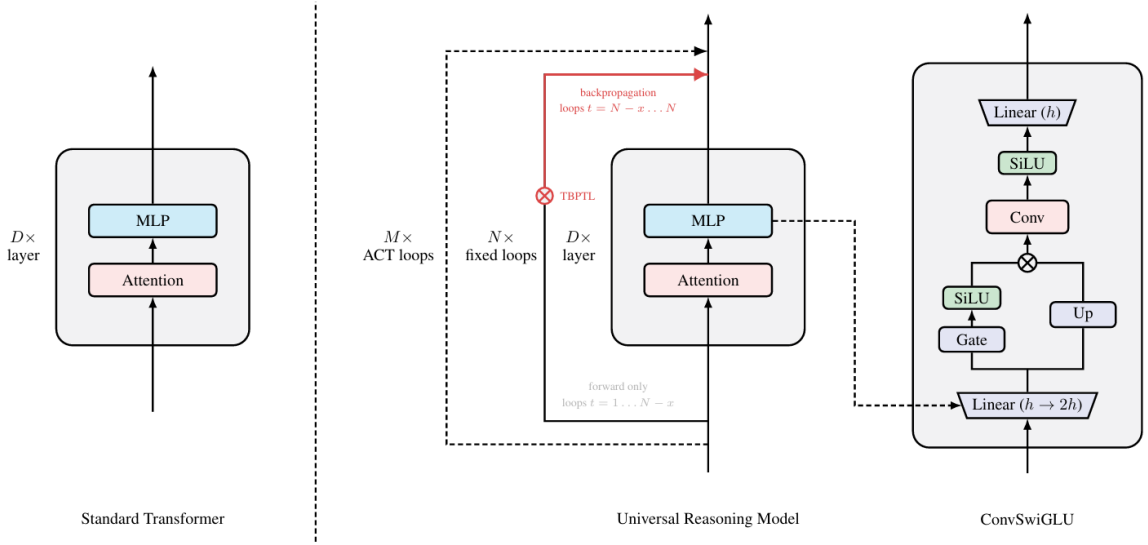
⟹ **TBPTL**

	ARC-AGI 1				ARC-AGI 2				Sudoku
	pass@1	pass@10	pass@100	pass@1000	pass@1	pass@10	pass@100	pass@1000	pass@1
HRM	34.4	46.4	55.0	60.5	5.4	9.6	14.3	18.6	63.9
TRM	40.0	51.3	59.8	64.4	4.6	7.4	11.7	13.6	66.8
URM	**53.8**	**71.3**	**80.4**	**85.1**	**16.0**	**26.9**	**34.3**	**41.3**	**77.6**
w/o Short Conv.	45.3	62.6	72.0	78.3	-	-	-	-	-
w/o Trunc. Backprop.	40.0	54.4	64.5	70.5	-	-	-	-	-

[2] Gao, Z., et al. (2025). *Universal Reasoning Model*.

Universal Reasoning Model (URM)

Full Architecture



Model	Parameter Size	ARC-AGI 1 (pass@1)	ARC-AGI 1 (pass@1000)
***Standard Transformers** (32 layers, 512 hidden)	~64M	23.75%	43.38%
***Standard Transformers** (4 layers, 512 hidden)	~8M	5.13%	12.63%
***URM** (Universal Reasoning Model)	~15M	53.8%	85.1%

Table 1: ARC-AGI performance comparison.

* trained from scratch on task-specific data, no pre-training

[2] Gao, Z., et al. (2025). *Universal Reasoning Model*.

Universal Transformers (UT) – **Caveats**

Important caveats to keep in mind:

These are **task-specific small models**, trained specifically for ARC-AGI-type problems

- *When tested on fundamental sequence manipulation tests (such as copying strings, reversing tokens, or retrieving the first/last token), the URM fails completely, often scoring near 0%

However:

- Within a puzzle task they do show signs of reasoning oppose to memorization
- Achieves strong results at a **fraction of model complexity**

[5] third-party evaluation by Aragon (2025) (richardaragon.substack.com)

My thoughts

The Good

- Main focus of the paper is optimizing UT architecture: Genuinely useful ablations (Insights may inform future architectures at scale)
- Fair comparisons of UT variants
- Clean codebase & experiments

The Bad

- *Inefficient training: The recurrent structure creates complex loss landscapes that fail to converge using industry-standard optimizers like AdamW

  → end up **training exponentially slower** than standard transformers despite having fewer parameters

- “Recurrence beats brute-force scaling when it comes to reasoning” - was the original UT claim (not specific to this paper)…
  - Neither proven nor disproven
  - URM and UTs are **far behind frontier models** on broader reasoning benchmarks

[5] third-party evaluation by Aragon (2025) (richardaragon.substack.com)

My thoughts

Open Points

- We have not seen tests at scale (are UTs generalizable, scalable reasoners?)
- Could UT-style recurrence combine with pretraining meaningfully?
  1. Add internet scale pre-training to UTs
  2. Add a recurring reasoning module to a vanilla LLM transformer for hard subproblems