Less is More: Recursive Reasoning with Tiny Networks

Alexia Jolicoeur-Martineau
Samsung SAIL Montréal
arXiv:2510.04871

Xi-Wei Pan Oct.15, 2025

TL; DR

- A single tiny network **recursively** refines its latent reasoning state to improve answers **step by step**—achieving strong generalization through thinking depth rather than network depth.

- Why recursion helps so much remains to be explained (overfitting?).

- A supervised learning method rather than generative model.

Contents

System 1 & System 2 Thinking

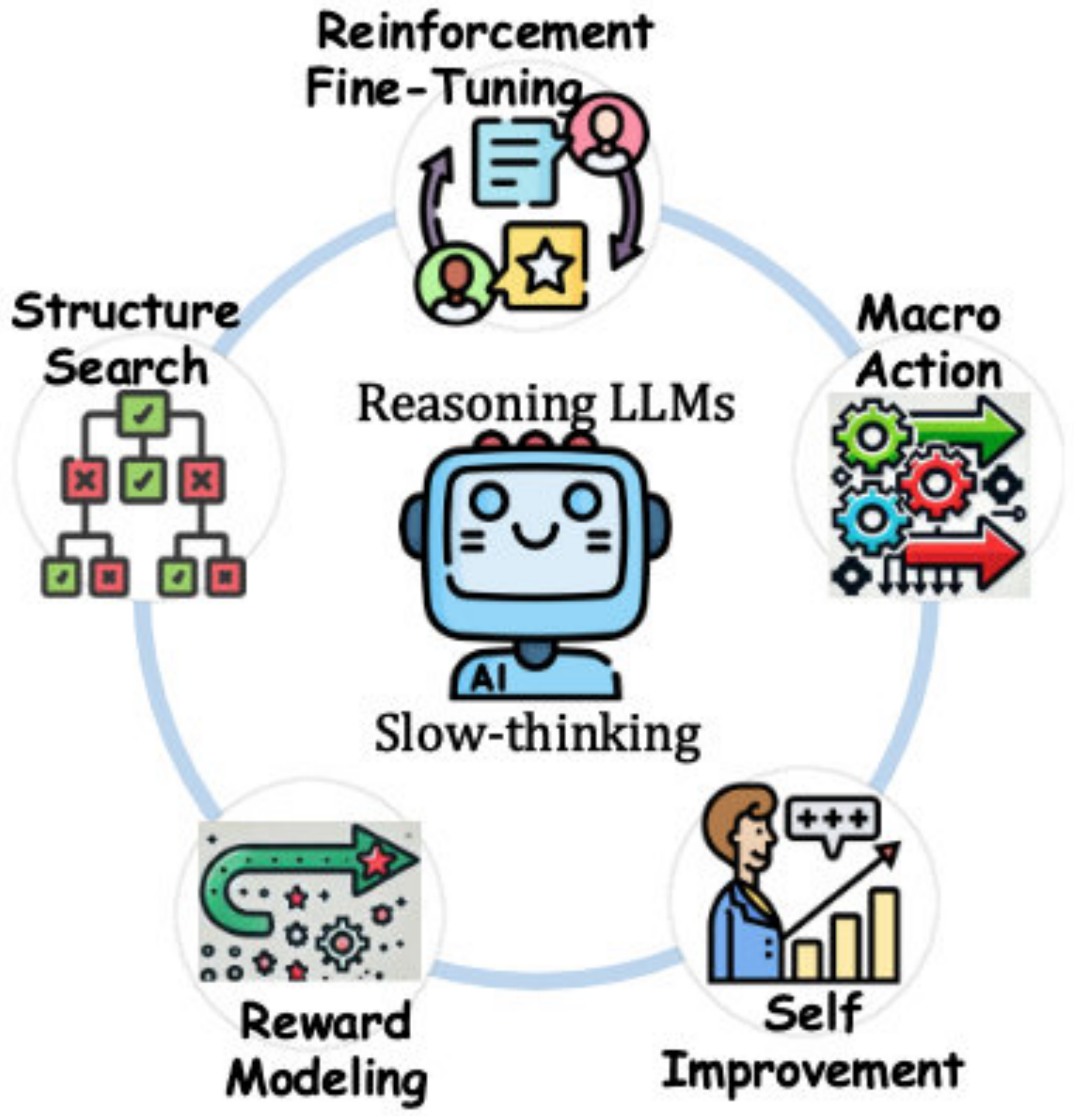
- Two different modes of cognitive processing
- **System 1 is fast, automatic, intuitive, and emotional.**
  effortlessly and quickly,
  guides our daily decisions, judgments, and impressions.
  eg. Feed-forward Neural Network, Transformer?
- **System 2 is slow, deliberate, and analytical.**
  It is activated when we need to perform complex computations.
  eg. expert-level coding, competitive math, and PhD-level science questions.
  LLM: Not Turing-complete, hard to do algorithmic reasoning.

LLM + …

Kahneman, D. (2011). Thinking, fast and slow. macmillan.

From Next-Token Prediction to Reasoning

- Traditional next-token prediction model: $p\left(x_t|\mathbf{x}_{<t}\right) \leftarrow$ No explicit reasoning
- **Reasoning:** devising and executing complex goal oriented action sequences
  - The **Chain of Thought (CoT)** asks model to "think step by step".
  - The **Test-Time Compute (TTC)** lets model "think more" before answering.



The reasoning abilities in LLMs are typically developed through **alignment**, including

- supervised fine-tuning (SFT),
- reinforcement learning (RL).

Li, et al. (2025). From system 1 to system 2: A survey of reasoning large language models. arXiv:2502.17419.

Deep Equilibrium Model

Motivation - Beyond Layer Stacking

- Deep neural networks (like Transformers) rely on stacking layers for better performance.

$$z^i = f_\theta^{i-1}(z^{i-1}, \tilde{x}), \text{ where } i = 1, 2, \cdots, L$$

- But increasing depth → vanishing gradients, huge memory cost, slow inference.

$$f_\theta^i = f_\theta, \ \forall i$$

- Q: If the same transformation is applied at each layer of a deep network, what is the limit of this process?

Bai, et al. (2019). Deep equilibrium models. *Advances in neural information processing systems*, *32*.

Deep Equilibrium Model

Mathematical Formulation

- Instead of explicitly computing multiple layers, a DEQ finds a **fixed point**:

$$z^* = f_\theta(z^*, \tilde{x})$$

- Here, z^* is the representation **at equilibrium**, meaning applying further does not change z^*.

- Unlike a conventional network where the output is the activations from the L^{th} layer, **the output of a DEQ is the equilibrium point itself**.

Bai, et al. (2019). Deep equilibrium models. *Advances in neural information processing systems*, *32*.

Deep Equilibrium Model

Finding the Equilibrium: Root-Finding Methods

- Fixed-Point Iteration: $z^{t+1} = f_\theta(z^t, \tilde{x})$ Slow!
- Broyden's Method (Quasi-Newton)

$$g_\theta(z, \tilde{x}) \coloneqq f_\theta(z, \tilde{x}) - z \to 0$$

$$z^{t+1} = z^t - J_g^{-1} g(z^t)$$

- Any root-find method is fine.

Bai, et al. (2019). Deep equilibrium models. *Advances in neural information processing systems*, *32*.

Deep Equilibrium Model

Training DEQs with Implicit Differentiation

- Unlike traditional deep networks that require storing activations for back-propagation, DEQs train using **implicit differentiation**.
- Loss function: $\mathcal{L} = \mathcal{L}(z^*, y)$ $\quad \frac{d\mathcal{L}}{d\theta} = \frac{\partial \mathcal{L}}{\partial z^*} \frac{dz^*}{d\theta}$
- **Implicit Function Theorem** $\quad g_\theta(z, \tilde{x}) := f_\theta(z, \tilde{x}) - z$

$$\frac{dg}{d\theta} = \frac{\partial f}{\partial \theta} + \frac{\partial f}{\partial z} \frac{dz^*}{d\theta} - \frac{dz^*}{d\theta} = 0$$

$$(I - J_f) \frac{dz^*}{d\theta} = \frac{\partial f}{\partial \theta}$$

Hierarchical Reasoning Model

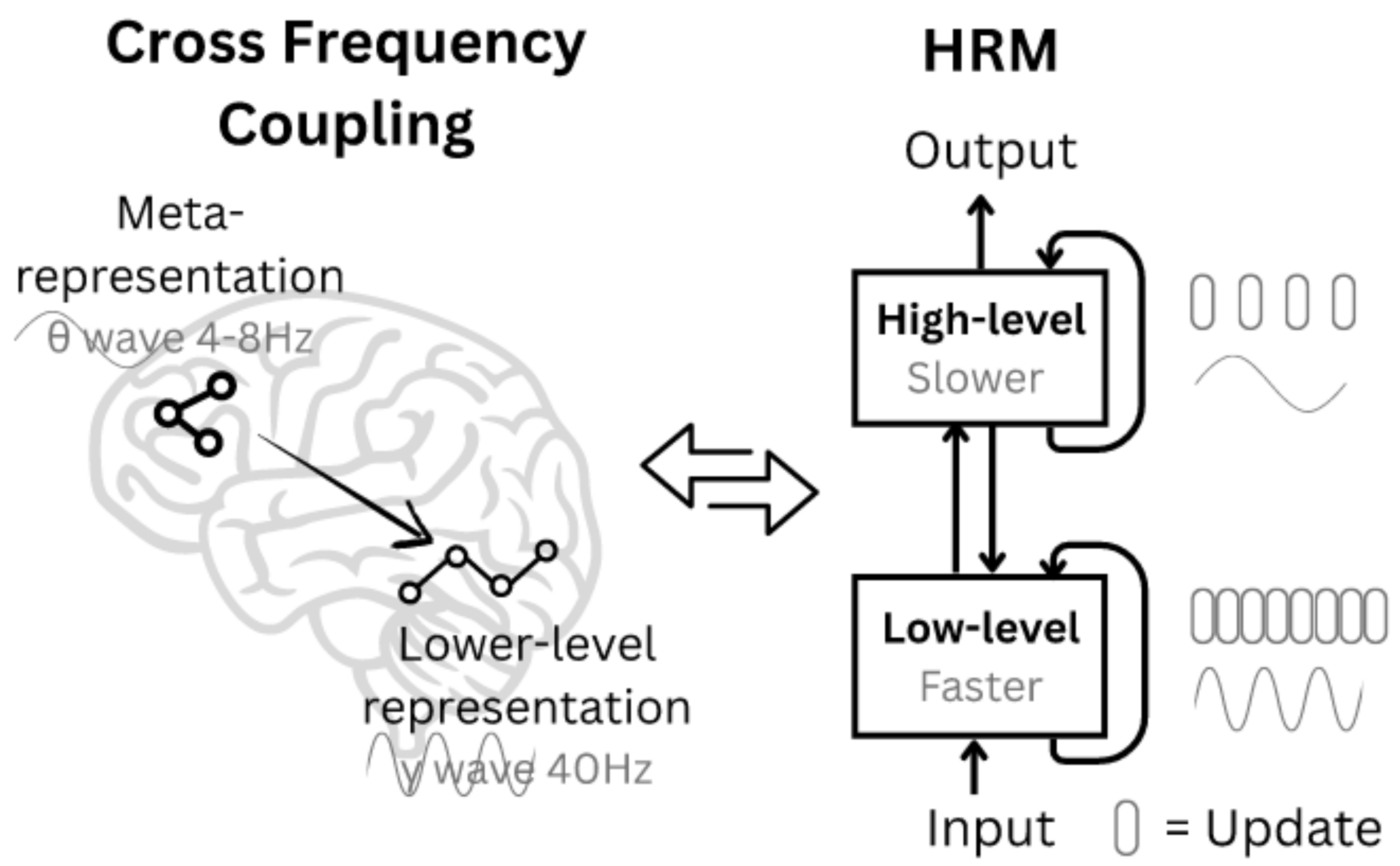
Motivation

Reasoning: devising and executing complex goal oriented action sequences

Inspiration from the brain: signals at different frequencies + recurrence

Gap in previous work:

- CoT suffer from brittle task decomposition, high data and compute requirements
- (Transformers don't scale well)



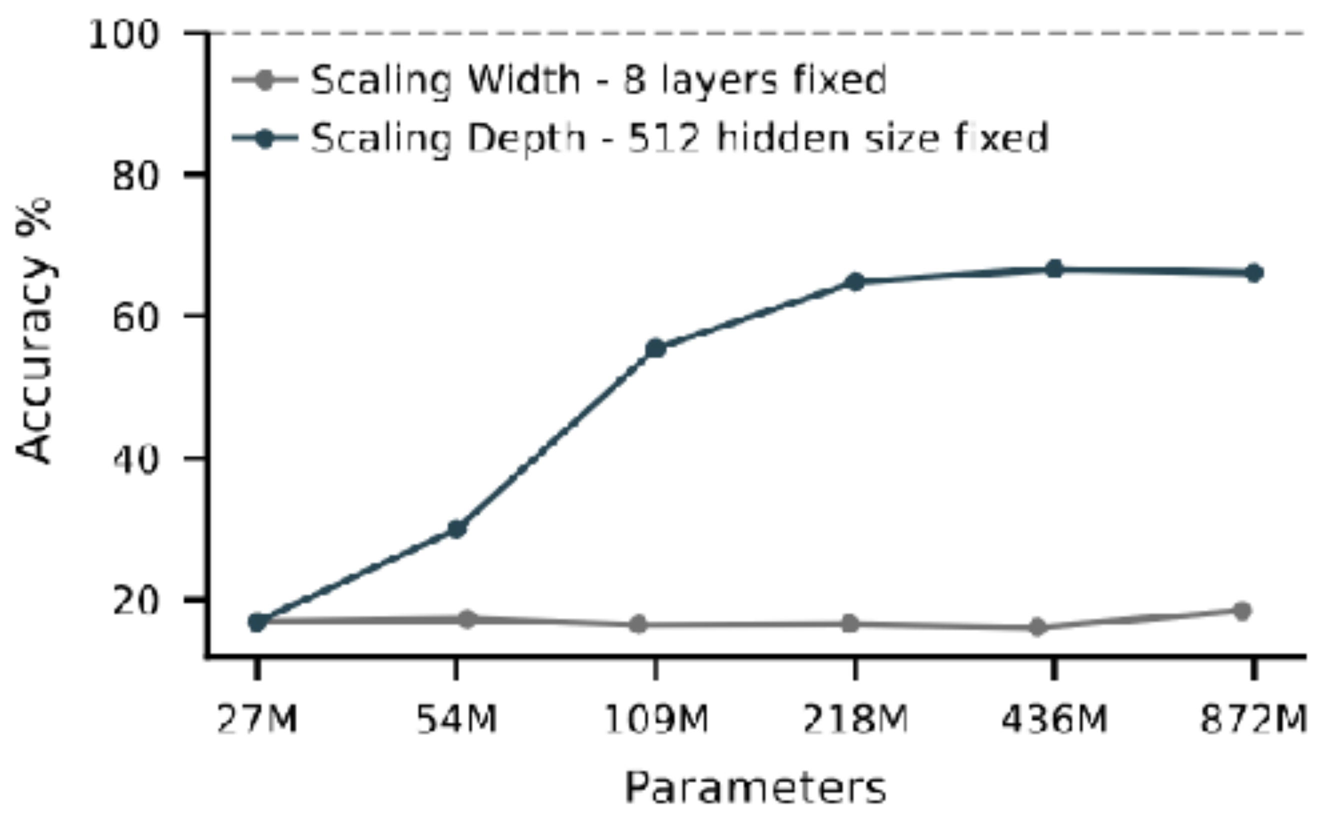


Image Source: Wang, et al. (2025). Hierarchical Reasoning Model. *arXiv:2506.21734*.

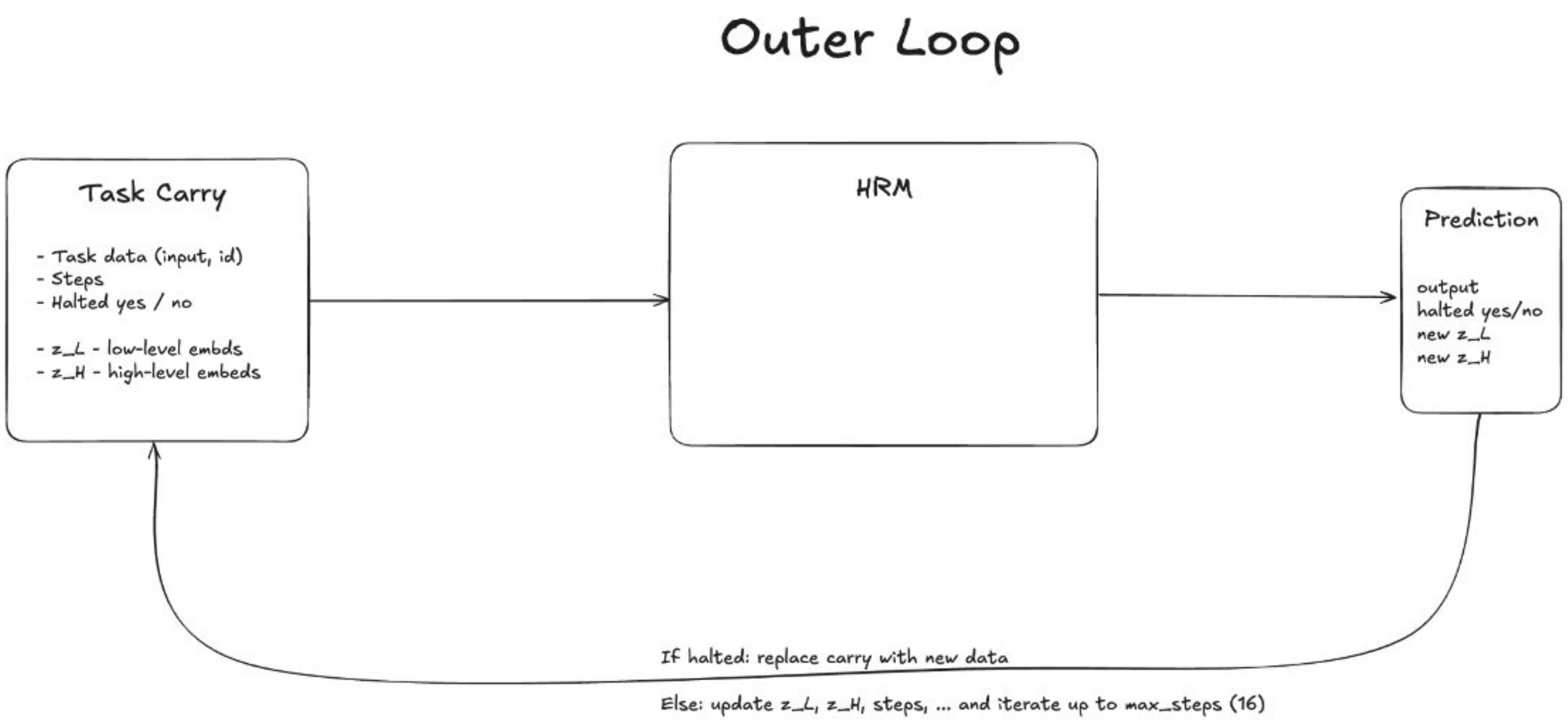
Hierarchical Reasoning Model

Overview

- Paper: https://arxiv.org/abs/2506.21734
- Code: https://github.com/sapientinc/HRM
- Analysis blog: https://arcprize.org/blog/hrm-analysis
- Analysis code: https://github.com/arcprize/hierarchical-reasoning-model-analysis

Model Architecture

Hierarchical Reasoning Model - Methods



Iterative Refinement

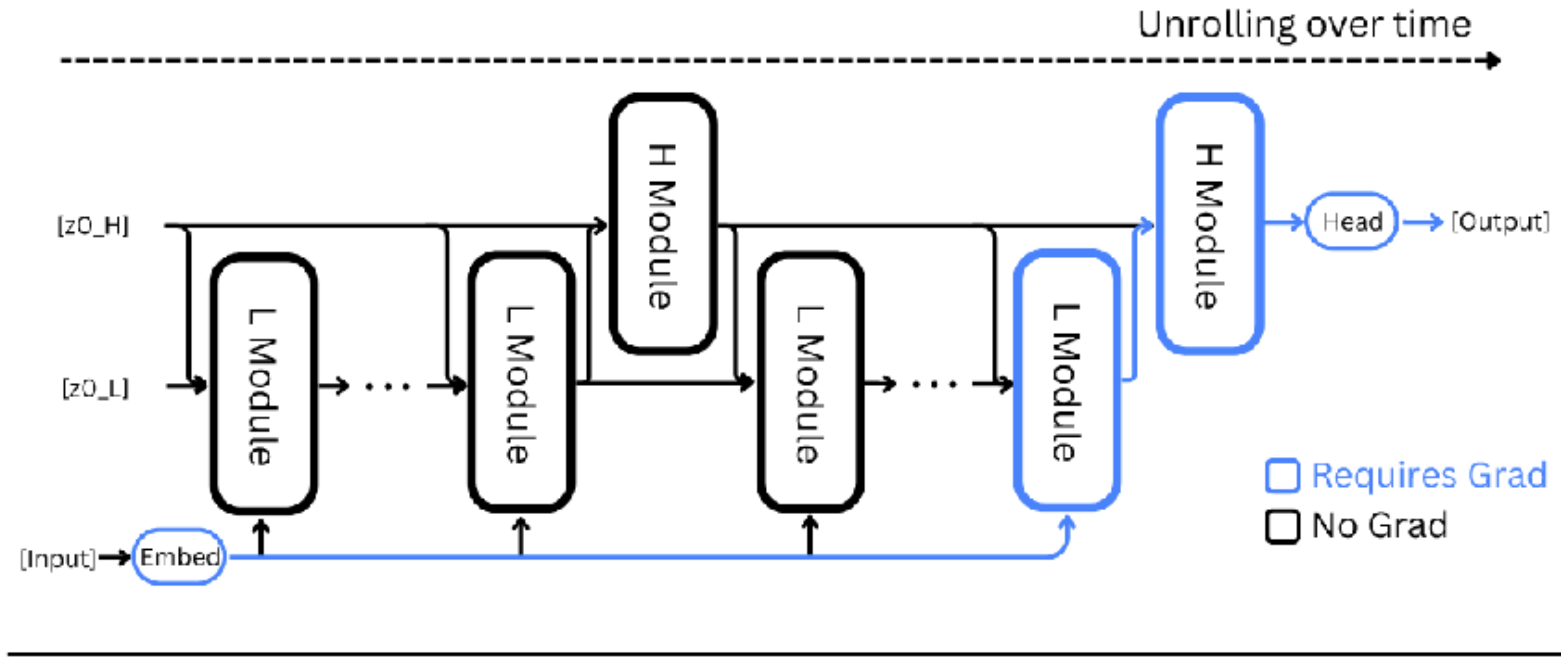
- given a context + initial guess, refine towards equilibrium
- Outer loop +
  2 levels of inner loop
- Deep supervision (segment)

Image Source: https://docs.google.com/presentation/d/12IAuVKZXvbW6uCwzDhzN1PBh4fdjypjqyucYzoKJKMg/edit?usp=sharing

Model Architecture

Hierarchical Reasoning Model - Methods

$$\tilde{x} = f_I(x; \theta_I) \qquad \hat{y} = f_O(z_H^{NT}; \theta_O)$$




```
def hrm(z, x, N=2, T=2):
    x = input_embedding(x)
    zH, zL = z

    with torch.no_grad():
        for _i in range(N * T - 1):
            zL = L_net(zL, zH, x)
            if (_i + 1) % T == 0:
                zH = H_net(zH, zL)

    # 1-step grad
    zL = L_net(zL, zH, x)
    zH = H_net(zH, zL)
    return (zH, zL), output_head(zH)
```

```
# Deep Supervision
for x, y_true in train_dataloader:
    z = z_init
    for step in range(N_supervision):
        z, y_hat = hrm(z, x)

        loss = softmax_cross_entropy(y_hat, y_true)
        z = z.detach()

        loss.backward()
        opt.step()
        opt.zero_grad()
```

Low-level: T time-steps each cycle

$$z_L^i = f_L(z_L^{i-1}, z_H^{i-1}, \tilde{x}; \theta_L)$$

High-level: executed once per cycle for N cycle

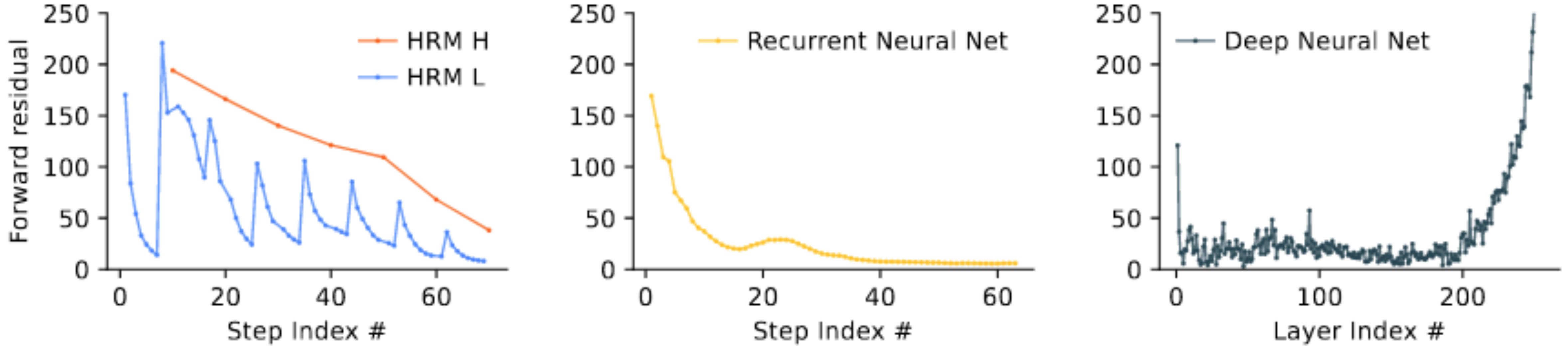
$$z_H^i = \begin{cases} f_H(z_L^{i-1}, z_H^{i-1}, \tilde{x}; \theta_L) & \text{, if } i \equiv 0 \pmod{T}; \\ z_H^{i-1} & \text{, otherwise.} \end{cases}$$

Image Source: Wang, et al. (2025). Hierarchical Reasoning Model. *arXiv:2506.21734*.

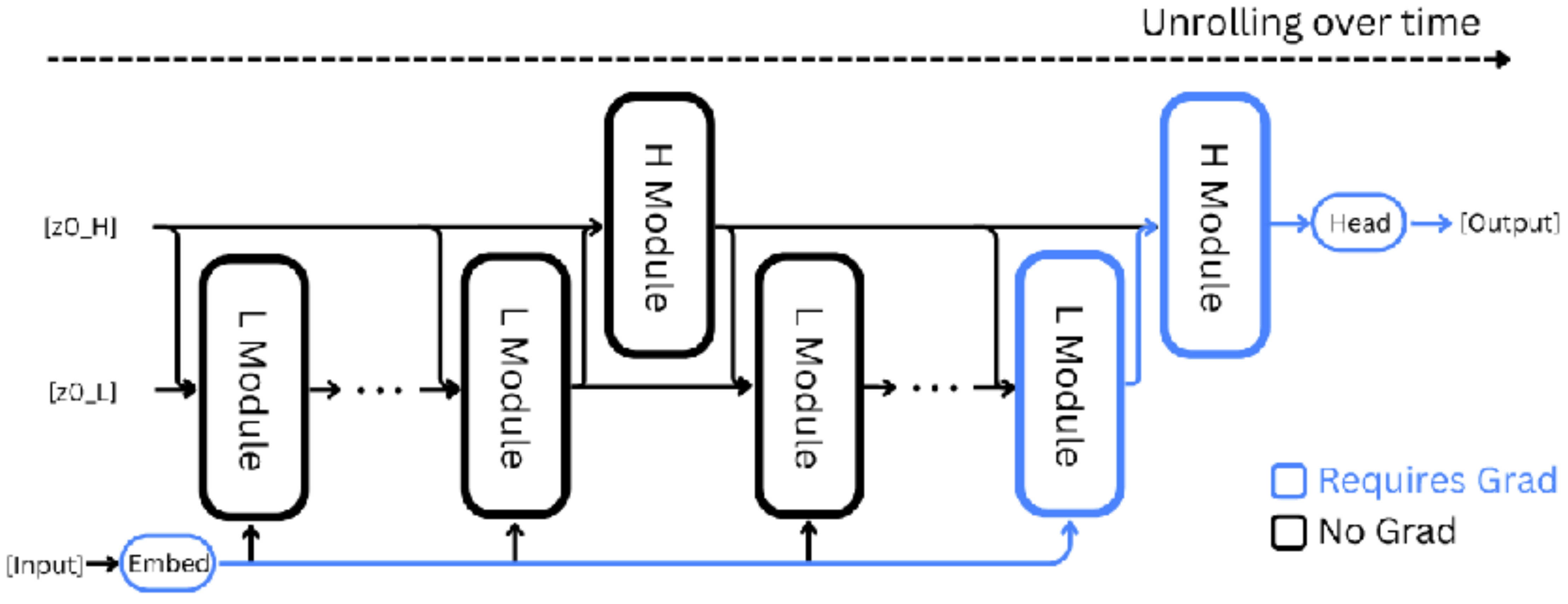
Image Source: https://docs.google.com/presentation/d/12IAuVKZXvbW6uCwzDhzN1PBh4fdjypjqyucYzoKJKMg/edit?usp=sharing

Approximate gradient

Hierarchical Reasoning Model



$$L^m_{\mathrm{ACT}} = \mathrm{Loss}(\hat{y}^m, y) + \mathrm{BinaryCrossEntropy}(\hat{Q}^m, \hat{G}^m)\ .$$



$$(I - J_f)\frac{dz^*}{d\theta} = \frac{\partial f}{\partial \theta}$$

$$(I - J_f)^{-1} = I + J_f + J_f^2 + \cdots$$
$$\approx I$$

One-Step gradient approximation

$$\frac{\partial z_H^*}{\partial \theta_H} \approx \frac{\partial f_H}{\partial \theta_H}, \quad \frac{\partial z_H^*}{\partial \theta_L} \approx \frac{\partial f_H}{\partial z_L^*} \cdot \frac{\partial z_L^*}{\partial \theta_L}, \quad \frac{\partial z_H^*}{\partial \theta_I} \approx \frac{\partial f_H}{\partial z_L^*} \cdot \frac{\partial z_L^*}{\partial \theta_I}\ . \qquad \frac{\partial z_L^*}{\partial \theta_L} \approx \frac{\partial f_L}{\partial \theta_L}, \quad \frac{\partial z_L^*}{\partial \theta_I} \approx \frac{\partial f_L}{\partial \theta_I}\ .$$

Image Source: Wang, et al. (2025). Hierarchical Reasoning Model. *arXiv:2506.21734*.

Adaptive computational time (ACT)

Hierarchical Reasoning Model - Methods

$$L_{\text{ACT}}^{m} = \textsc{Loss}(\hat{y}^{m}, y) + \textsc{BinaryCrossEntropy}(\hat{Q}^{m}, \hat{G}^{m}) \ .$$

- **Q-learning: Markov Decision Process (MDP)**
- action = {halt, continue}, state = $\{z_H^m\}$
- Reward: halt → 1 if predicts correct; else 0. continue → 0.
- Q-value is predicted by a Q-head: $\hat{Q}^m = \sigma\left(\theta_Q^\top z_H^{mNT}\right) = \left(\hat{Q}^m_{\text{halt}}, \hat{Q}^m_{\text{continue}}\right)$
- According to Bellman Equation, the target Q-value function: $\hat{G}^m = \left(\hat{G}^m_{\text{halt}}, \hat{G}^m_{\text{continue}}\right)$

$$\hat{G}^m_{\text{halt}} = \mathbf{1}\{\hat{y}^m = y\}, \qquad \hat{G}^m_{\text{continue}} = \begin{cases} \hat{Q}^{m+1}_{\text{halt}}, & \text{if } m \geq N_{\max}, \\ \max\left(\hat{Q}^{m+1}_{\text{halt}},\ \hat{Q}^{m+1}_{\text{continue}}\right), & \text{otherwise.} \end{cases}$$

Wang, et al. (2025). Hierarchical Reasoning Model. *arXiv:2506.21734*.

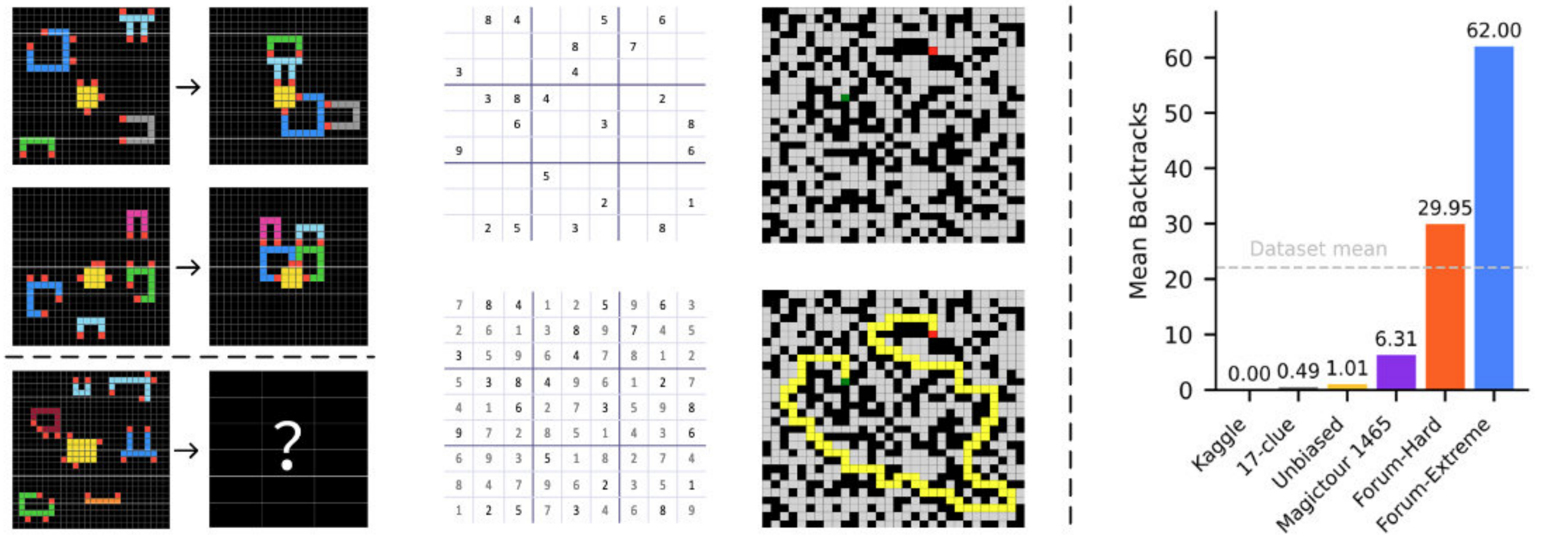
Method Summary

Hierarchical Reasoning Model

- **Outer Refinement Loop:** Propose solution and understand when you're done -> adaptive computation
- **Inner Equilibrium Loop:** 'hierarchical' refinement towards equilibrium
- **Puzzle Embeddings:** learn to associate a puzzle hash with a specific transformation

Benchmarks

Hierarchical Reasoning Model



(a) ARC-AGI (b) Sudoku-Hard (c) Maze navigation (d) *Sudoku-Extreme* subset difficulty

Image Source: Wang, et al. (2025). Hierarchical Reasoning Model. *arXiv:2506.21734*.

Data Augmentation

Hierarchical Reasoning Model

- Inference
  - Make separate predictions for all augmented versions of a task
  - Use maximum outer refinement loops, no ACT
  - Use majority voting on pool of predictions

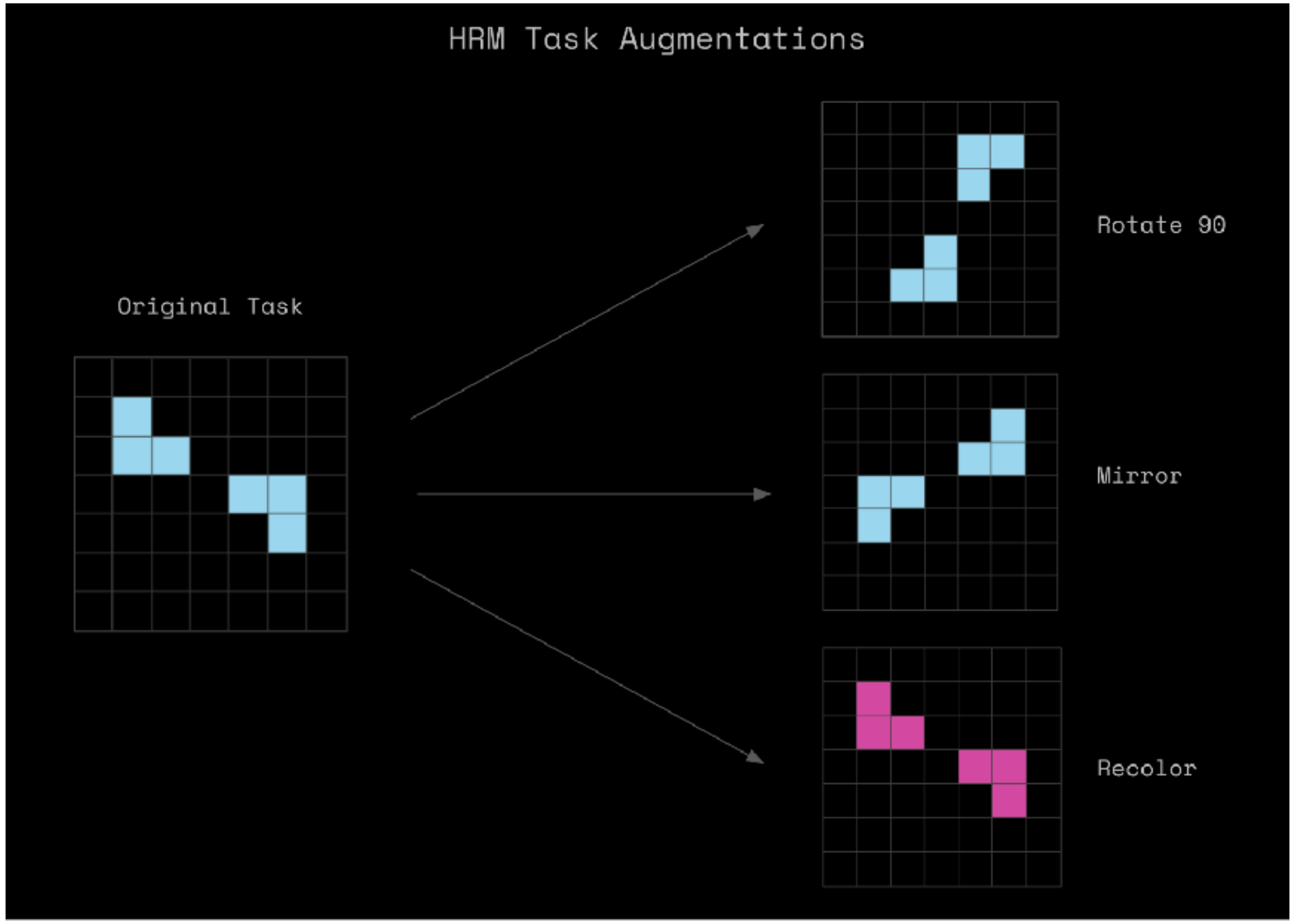


Image Source: https://arcprize.org/blog/hrm-analysis

Performance Overview

Hierarchical Reasoning Model

- Arc v1
    - Public 41% pass@2
    - Semi-Private 32% pass@2
- Arc v2
    - Public 4% pass@2
    - Semi-Private 2% pass@2

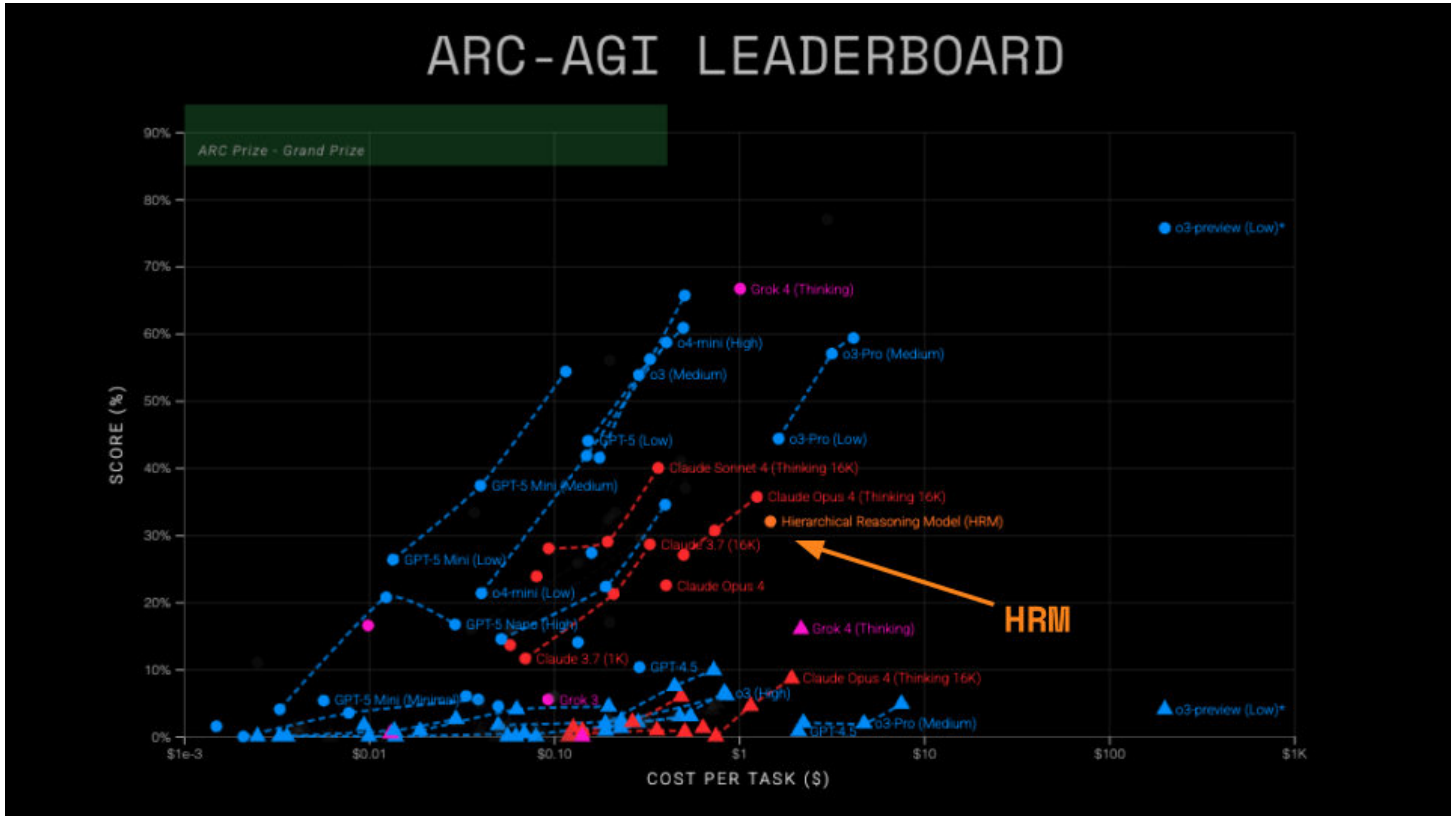


Image Source: https://arcprize.org/blog/hrm-analysis

Performance Overview

Hierarchical Reasoning Model

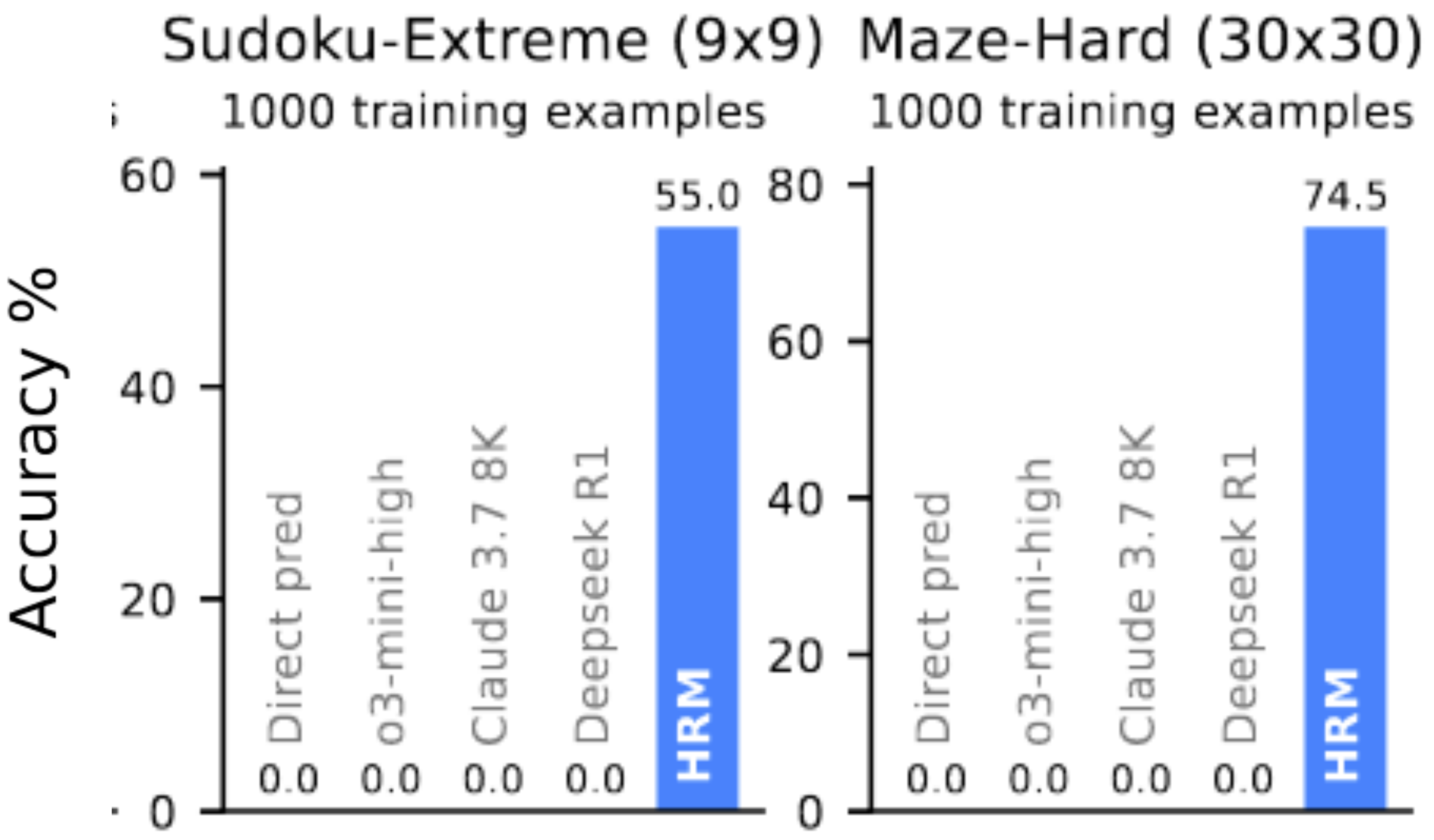


Image Source: Wang, et al. (2025). Hierarchical Reasoning Model. *arXiv:2506.21734*.

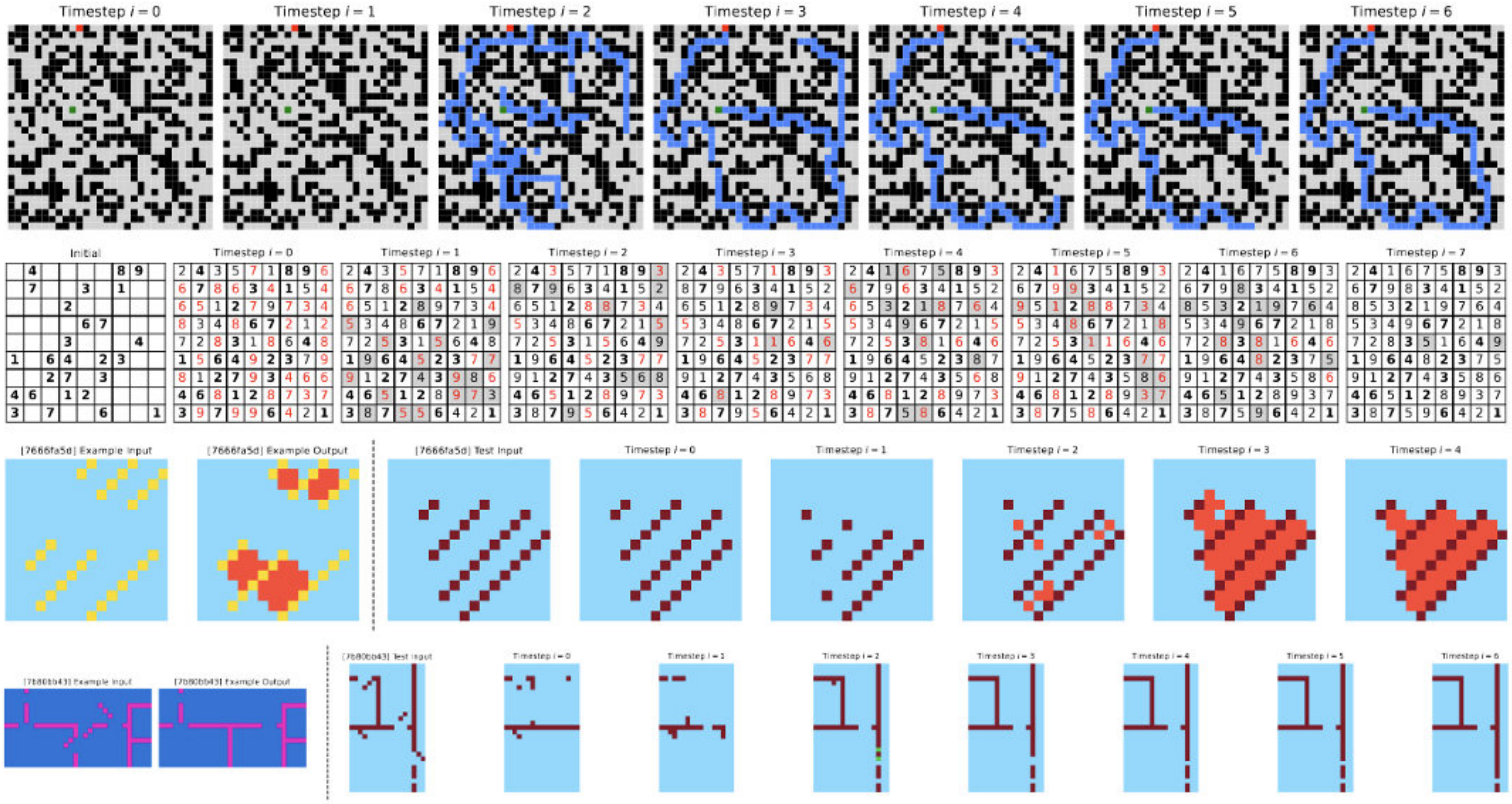


Image Source: Wang, et al. (2025). Hierarchical Reasoning Model. *arXiv:2506.21734*.

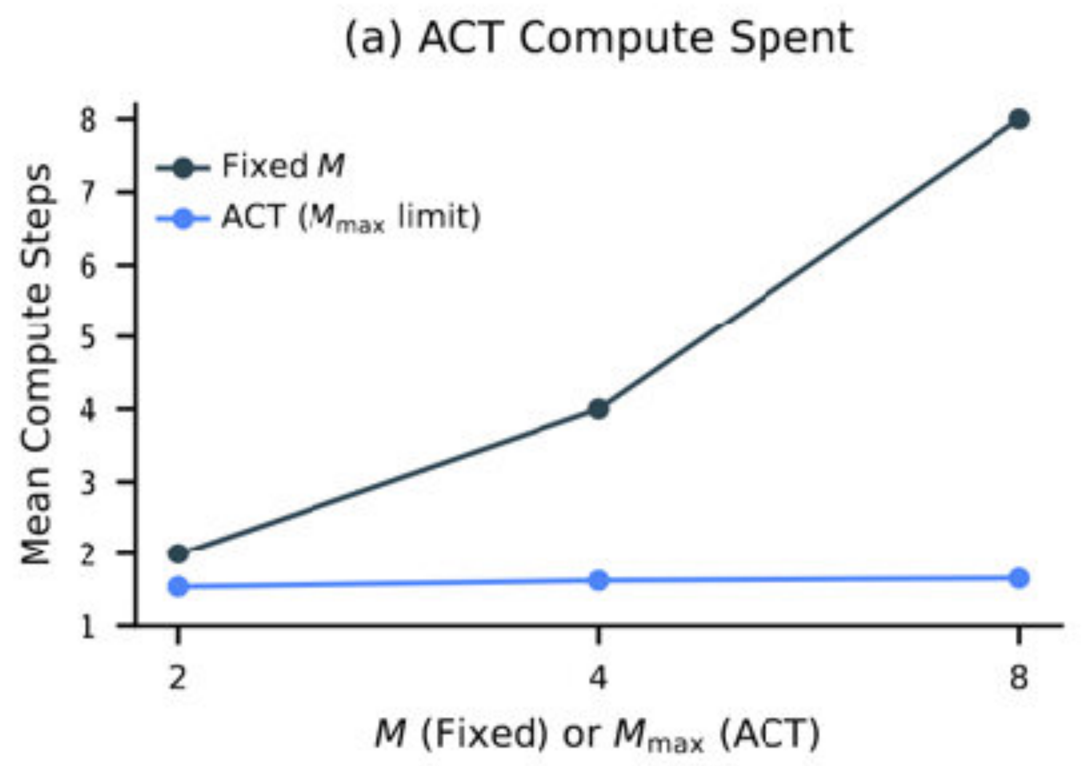
Why does it work?

Hierarchical Reasoning Model - Performance Drivers

- 1. Model architecture
- 2. Inner refinement loop
- 3. Outer refinement loop, with or without ACT
- 4. Data augmentation for HRM
- 5. Puzzle embeddings

Architecture?

Hierarchical Reasoning Model - Performance Drivers



- **Compare HRM with regular transformer**
  - Matched parameter count
  - Same hyperparameters
  - Less (forward) compute
- **Findings**:
  - Regular, unoptimized transformer is within ~5%.
  - No refinement -> matched
  - More outer loop steps -> gap closes

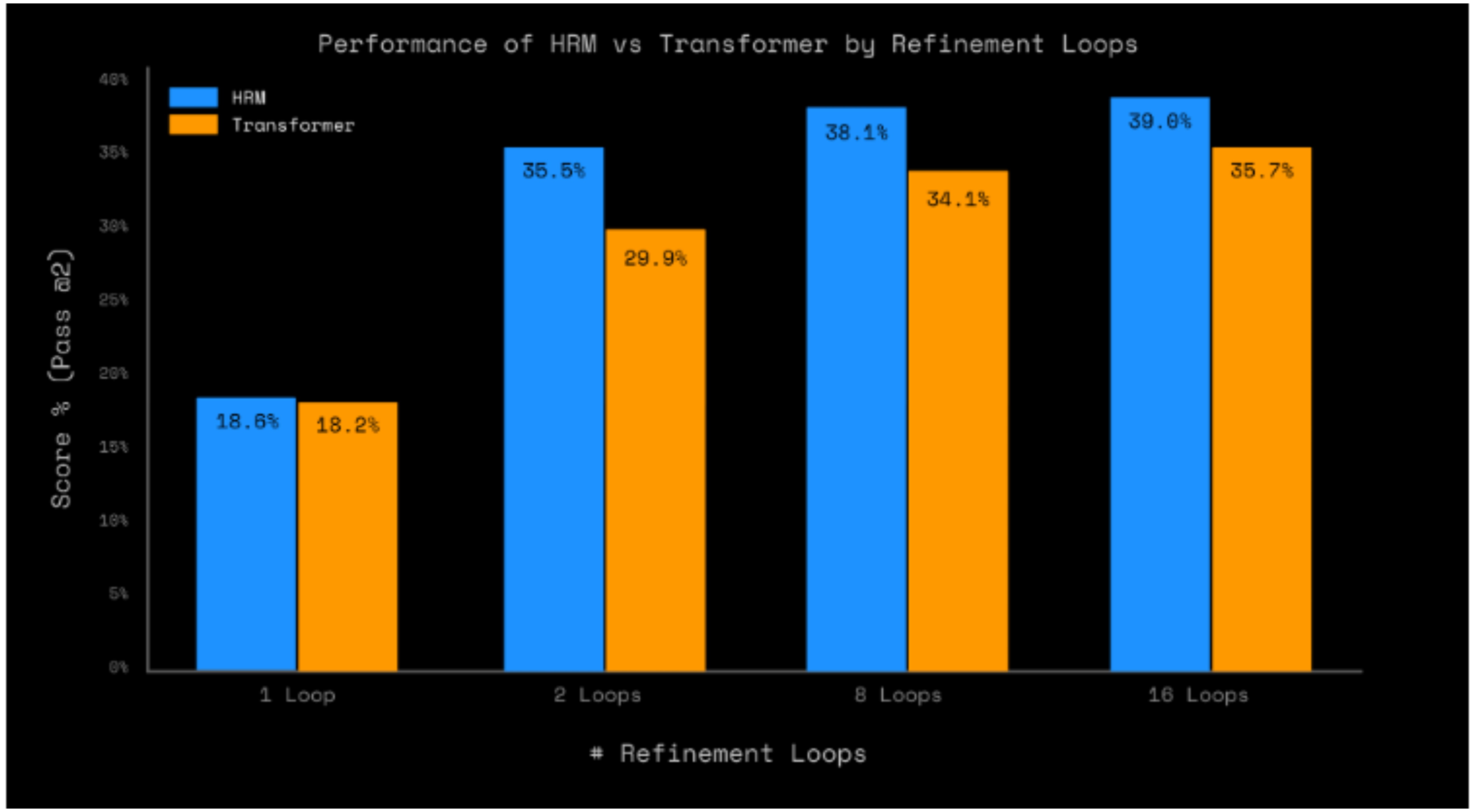
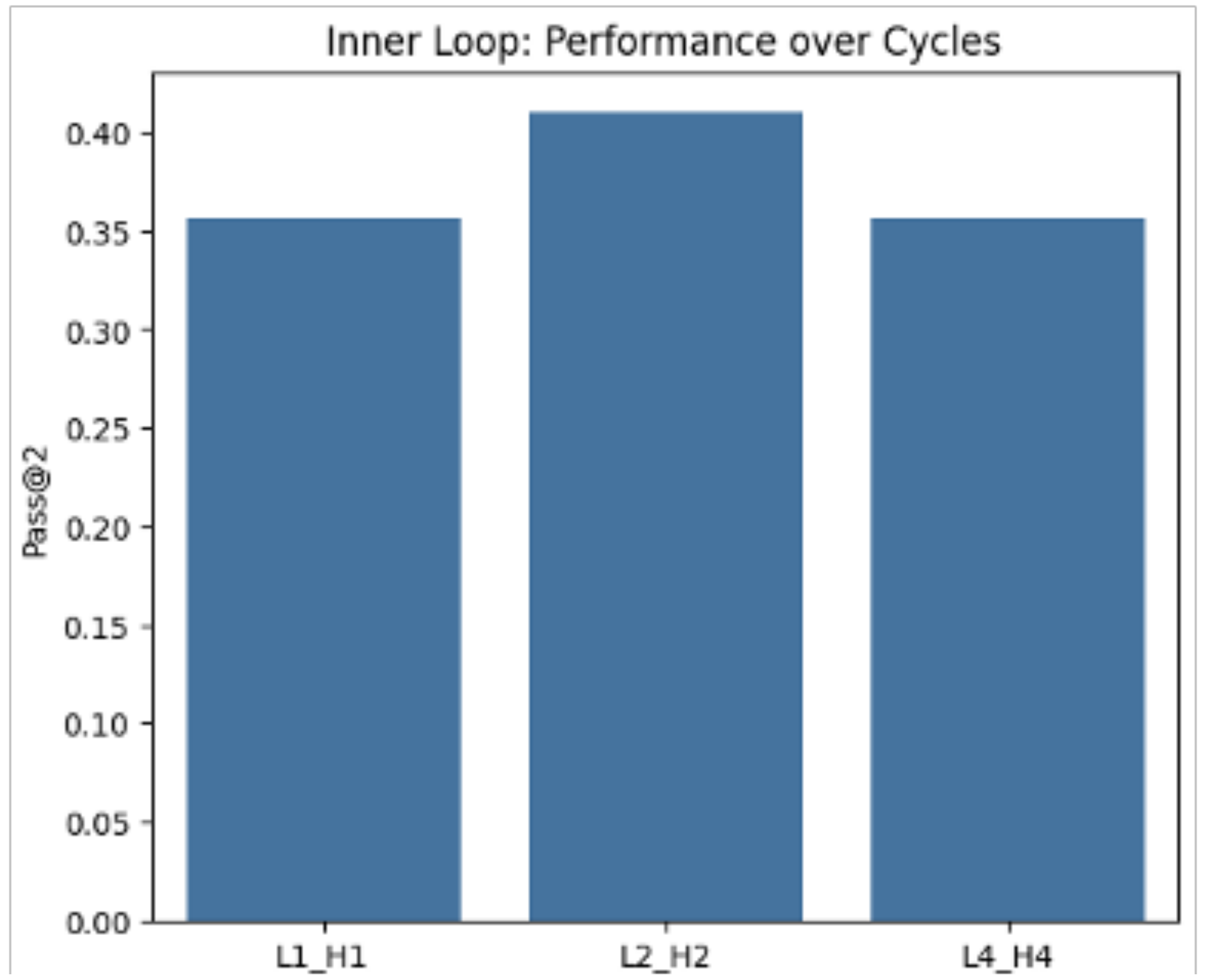


Image Source: https://arcprize.org/blog/hrm-analysis, Wang, et al. (2025). Hierarchical Reasoning Model. *arXiv:2506.21734*.

Inner Loop?

Hierarchical Reasoning Model - Performance Drivers

- Vary L and H cycles
- **Findings**
  - L1-H1 already achieves ⅞ of the performance.
  - Scaling higher does not improve further
  - #params remains the same, but compute scales with n_L*n_H.



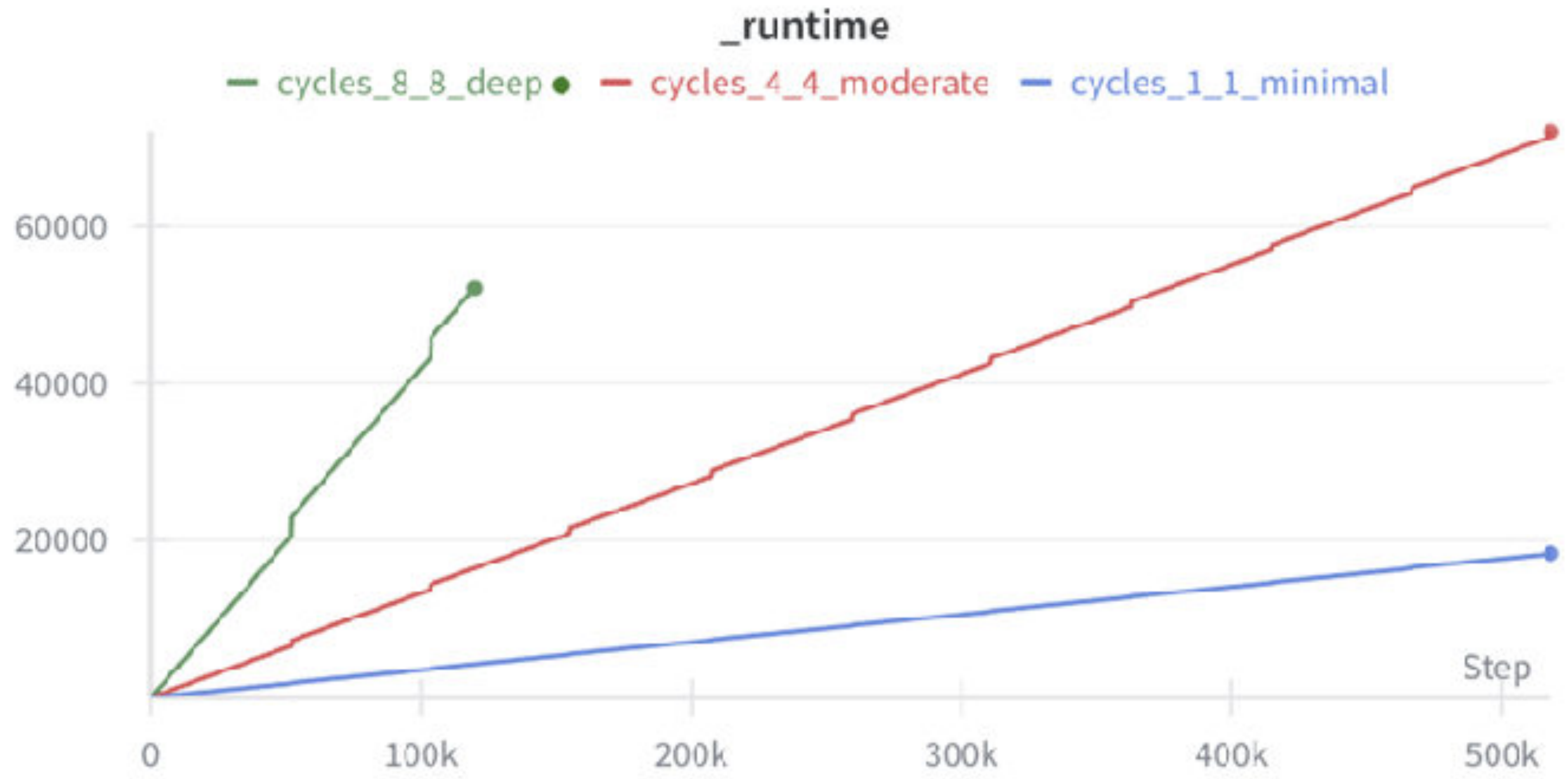
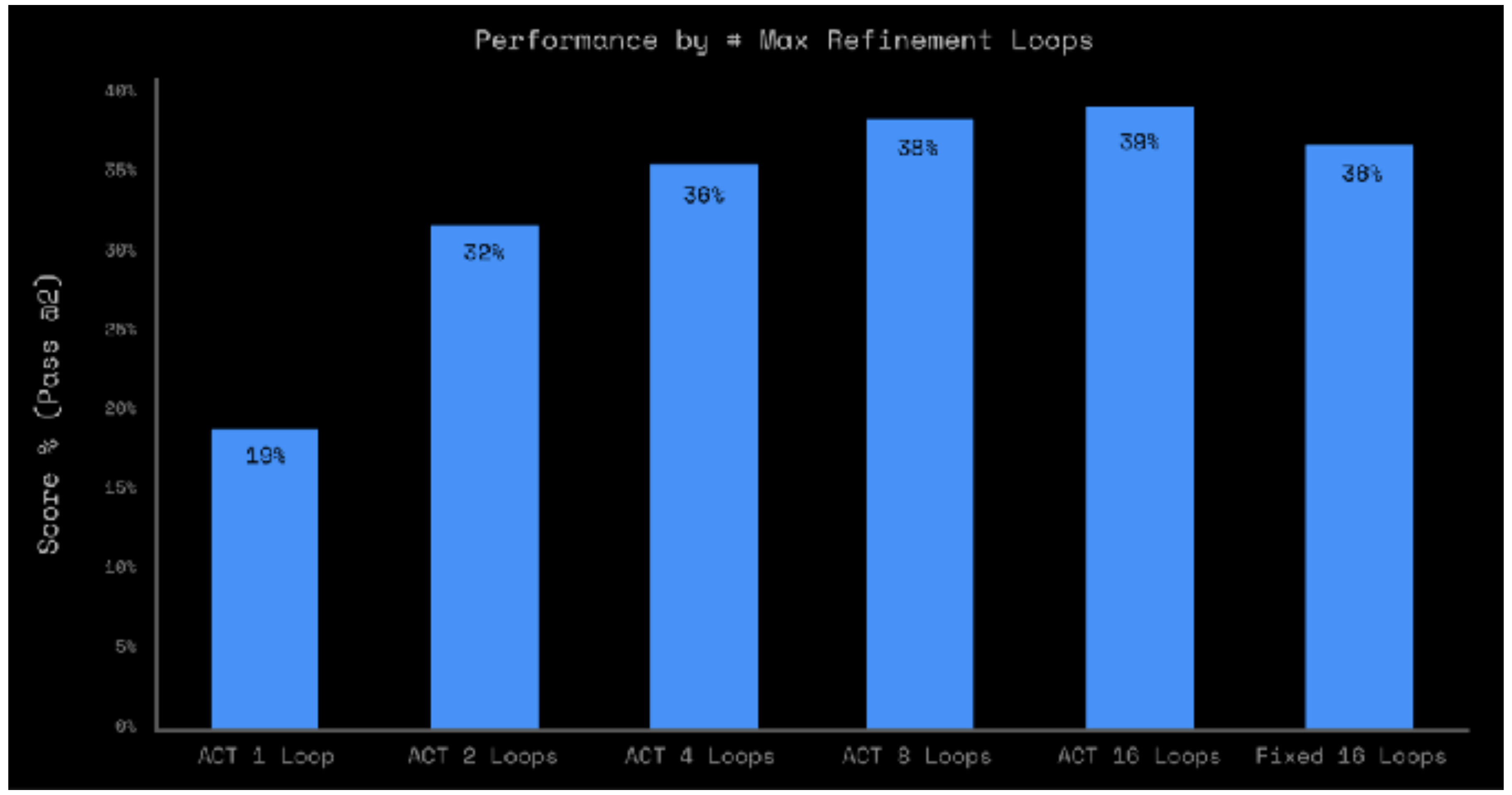


Image Source: https://arcprize.org/blog/hrm-analysis

Outer Loop + ACT?

Hierarchical Reasoning Model - Performance Drivers

- Vary max outer loop steps
- **Findings**
  - Refinement makes a big difference.
  - Refinement at training > refinement at inference
  - Learned halting signal leads works (only at training)
- **Refinement of predictions works!**



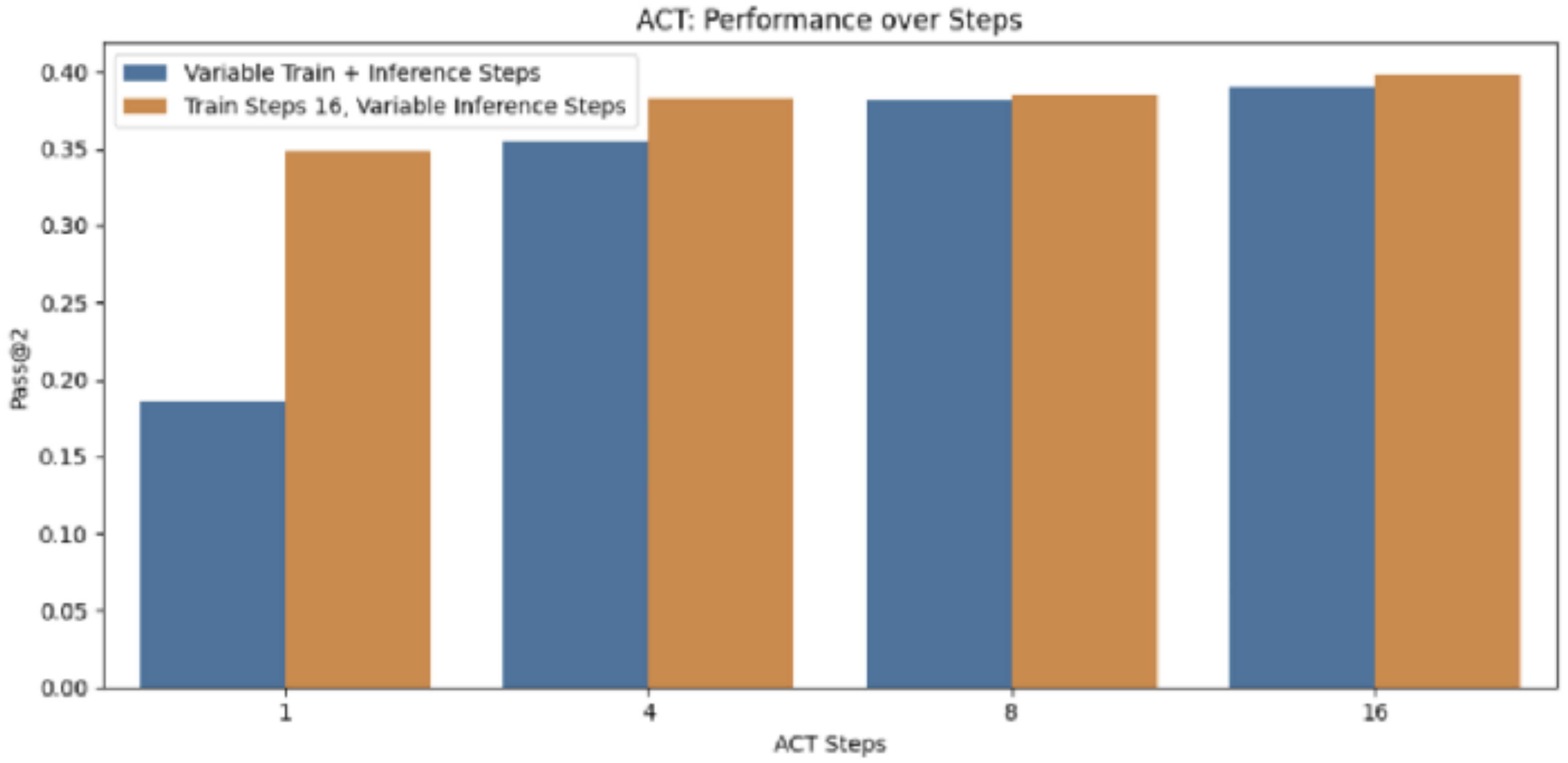


Image Source: https://arcprize.org/blog/hrm-analysis

Tiny Recursive Model

Overview

- Paper: https://arxiv.org/abs/2510.04871
- Code: https://github.com/SamsungSAILMontreal/TinyRecursiveModels

Motivations

Improvements in Hierarchical Reasoning Models

- Implicit Function Theorem with 1-step gradient approximation
  Although the residuals do generally reduce over time, a fixed point is unlikely to be reached when the theorem is actually applied.
- Twice the forward passes with Adaptive computational time (ACT)
- Why ‘Hierarchical’ interpretation? a bit forced…

Model Design

Tiny Recursive Model

$$z_L \leftarrow f_L(z_L + z_H + x)$$

$$\vdots$$

$$z_L \leftarrow f_L(z_L + z_H + x)$$

$$z_H \leftarrow f_H(z_L + z_H)$$

- No hierarchy! No IFT! No gradient approximation!
- Forget the biological story.
    - z_H = the current embedded solution. → Decoded via output head.
    - z_L = latent reasoning state, not directly a solution.
- Example: Sudoku —
    - z_H matches the current solution;
    - z_L does not.

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

Tokenized z_H (denoted y in TRM)

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

Tokenized z_L (denoted z in TRM)

Image Source: Jolicoeur-Martineau, A. (2025). Less is More: Recursive Reasoning with Tiny Networks. *arXiv:2510.04871*.

Model Design

Tiny Recursive Model

- Why 2 latent variables?
  a proposed solution y (z_H), and a latent reasoning feature z (z_L).

Table 2. TRM on Sudoku-Extreme comparing % Test accuracy when using more or less latent features

Method	# of features	Acc (%)
TRM y, z (Ours)	2	87.4
TRM multi-scale z	$n + 1 = 7$	77.6
TRM single z	1	71.9

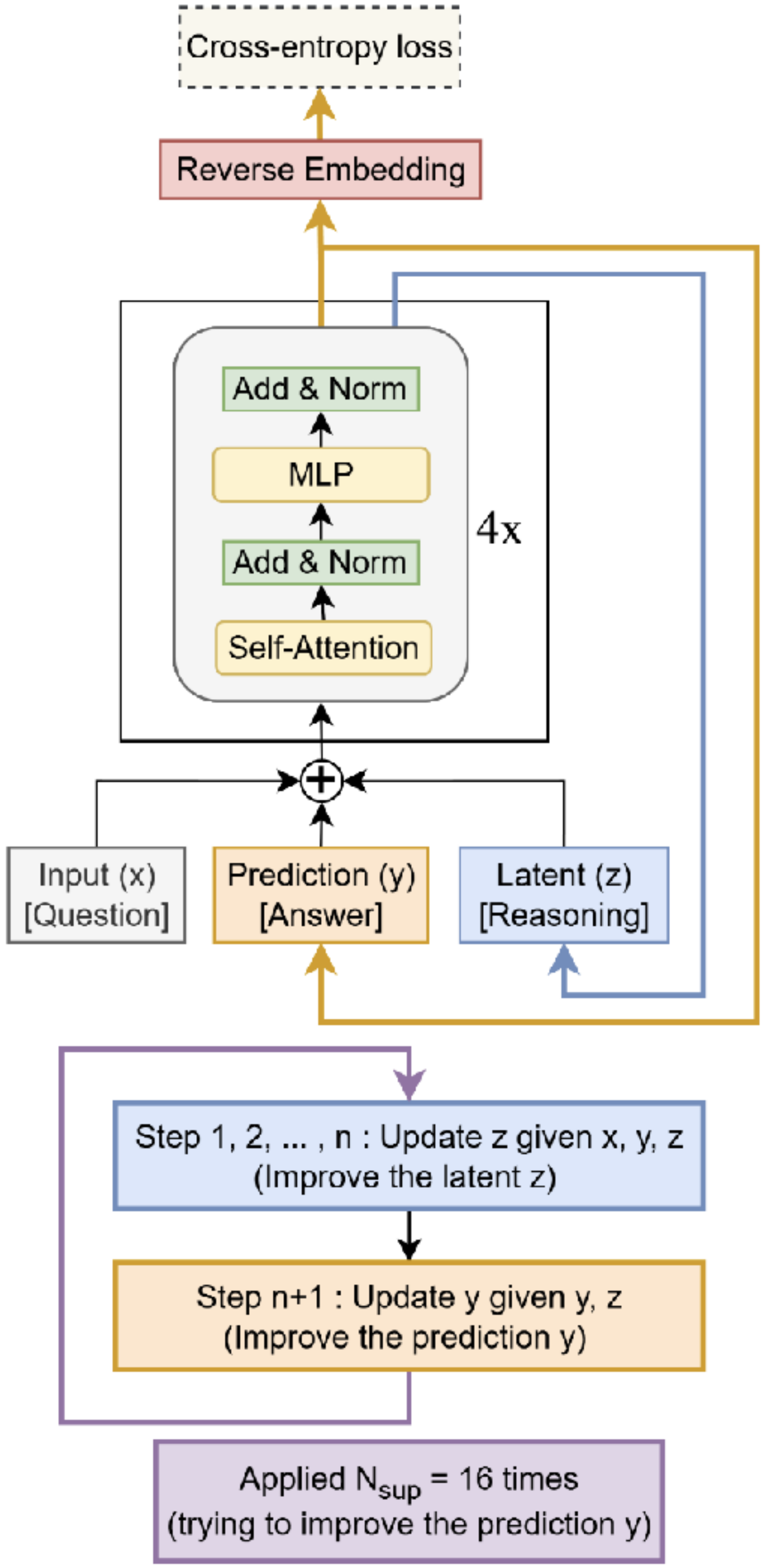


Image Source: Jolicoeur-Martineau, A. (2025). Less is More: Recursive Reasoning with Tiny Networks. *arXiv:2510.04871*.

```python
def latent_recursion(x, y, z, n=6):
    for i in range(n): # latent reasoning
        z = net(x, y, z)
    y = net(y, z) # refine output answer
    return y, z

def deep_recursion(x, y, z, n=6, T=3):
    # recursing T−1 times to improve y and z (no gradients needed)
    with torch.no_grad():
        for j in range(T-1):
            y, z = latent_recursion(x, y, z, n)
    # recursing once to improve y and z
    y, z = latent_recursion(x, y, z, n)
    return (y.detach(), z.detach()), output_head(y), Q_head(y)

# Deep Supervision
for x_input, y_true in train_dataloader:
    y, z = y_init, z_init
    for step in range(N_supervision):
        x = input_embedding(x_input)
        (y, z), y_hat, q_hat = deep_recursion(x, y, z)
        loss = softmax_cross_entropy(y_hat, y_true)
        loss += binary_cross_entropy(q_hat, (y_hat == y_true))
        loss.backward()
        opt.step()
        opt.zero_grad()
        if q_hat > 0: # early−stopping
            break
```

Model Design

Tiny Recursive Model

- Use single network
- Less is more: 2-layer-transformer
- No attention on Sudoku
- No ACT (Q learning)
- Iteration steps

Table 1. Ablation of TRM on Sudoku-Extreme comparing % Test accuracy, effective depth per supervision step ($T(n+1)n_{layers}$), number of Forward Passes (NFP) per optimization step, and number of parameters

Method	Acc (%)	Depth	NFP	# Params
HRM	55.0	24	2	27M
TRM ($T = 3, n = 6$)	87.4	42	1	5M
w/ ACT	86.1	42	2	5M
w/ separate f_H, f_L	82.4	42	1	10M
no EMA	79.9	42	1	5M
w/ 4-layers, $n = 3$	79.5	48	1	10M
w/ self-attention	74.7	42	1	7M
w/ $T = 2, n = 2$	73.7	12	1	5M
w/ 1-step gradient	56.5	42	1	5M

Table 3. % Test accuracy on Sudoku-Extreme dataset. HRM versus TRM matched at a similar effective depth per supervision step ($T(n+1)n_{layers}$)

		HRM $n = k$, 4 layers		TRM $n = 2k$, 2 layers	
k	T	Depth	Acc (%)	Depth	Acc (%)
1	1	9	46.4	7	63.2
2	2	24	55.0	20	81.9
3	3	48	61.6	42	87.4
4	4	80	59.5	72	84.2
6	3	84	62.3	78	OOM
3	6	96	58.8	84	85.8
6	6	168	57.5	156	OOM

Image Source: Jolicoeur-Martineau, A. (2025). Less is More: Recursive Reasoning with Tiny Networks. *arXiv:2510.04871*.

Performance

Tiny Recursive Model

Table 4. % Test accuracy on Puzzle Benchmarks (Sudoku-Extreme and Maze-Hard)

Method	# Params	Sudoku	Maze
Chain-of-thought, pretrained			
Deepseek R1	671B	0.0	0.0
Claude 3.7 8K	?	0.0	0.0
O3-mini-high	?	0.0	0.0
Direct prediction, small-sample training			
Direct pred	27M	0.0	0.0
HRM	27M	55.0	74.5
TRM-Att (Ours)	7M	74.7	85.3
TRM-MLP (Ours)	5M/19M[1]	87.4	0.0

Table 5. % Test accuracy on ARC-AGI Benchmarks (2 tries)

Method	# Params	ARC-1	ARC-2
Chain-of-thought, pretrained			
Deepseek R1	671B	15.8	1.3
Claude 3.7 16K	?	28.6	0.7
o3-mini-high	?	34.5	3.0
Gemini 2.5 Pro 32K	?	37.0	4.9
Grok-4-thinking	1.7T	66.7	16.0
Bespoke (Grok-4)	1.7T	79.6	29.4
Direct prediction, small-sample training			
Direct pred	27M	21.0	0.0
HRM	27M	40.3	5.0
TRM-Att (Ours)	7M	44.6	7.8
TRM-MLP (Ours)	19M	29.6	2.4

Image Source: Jolicoeur-Martineau, A. (2025). Less is More: Recursive Reasoning with Tiny Networks. *arXiv:2510.04871*.

Summary

Tiny Recursive Model

- Compared with HRM
  - Simpler: no fixed-point theorem, no biological hierarchy, no extra halting pass.
  - Smaller: one shared network → ≈ 1/4 the parameters of HRM.
  - More robust: better generalization on four reasoning benchmarks.

Conclusion

- A single tiny network **recursively** refines its latent reasoning state to improve answers **step by step**—achieving strong generalization through thinking depth rather than network depth.
- Why recursion helps so much remains to be explained (overfitting?).
- A supervised learning method rather than generative model.

Thanks!