ADAPTIVE SUPERVISION HELPS TINY RECURSIVE MODELS LEARN BETTER

Anonymous authors
Paper under double-blind review

ABSTRACT

Tiny recursive models (TRMs), as a class of latent depth-recursive models, demonstrate powerful reasoning capabilities and provide promising avenues for scaling reasoning capabilities in limited model size. Despite remarkable progress, training recursive models faces challenges in how supervision is applied across thinking steps. Supervising only the final output provides sparse learning signals, while TRMs address this by applying full supervision at every thinking step. However, at early thinking steps, full supervision can be noisy since the model may need more steps to derive a complete solution. Building on this intuition, we introduce AS-TRM, a simple yet powerful training method for TRMs with Adaptive Supervision. The core idea is to use the outcome of each thinking step to partition the training signals into focus and non-focus sets. We analyze AS-TRM through gradient and loss landscape diagnostics, and show that it reduces supervision noise and leads to faster convergence. Despite its simplicity, AS-TRM delivers substantial gains in both training speed and final performance: on Sudoku-Extreme, it achieves a 7.9-point accuracy gain (11.3% relative) and speeds up training by up to 1.79×; on Maze-Hard, AS-TRM enables flip and rotation augmentation, improving final accuracy by 6.3 points over an already strong baseline; and on ARC-AGI-1, it achieves up to 1.51× speedup.

1 INTRODUCTION

Recent work on Hierarchical Reasoning Models (HRMs) (Wang et al., 2025) and Tiny Recursive Models (TRMs) (Jolicoeur-Martineau, 2025), which we refer to collectively as *Latent Recursive Reasoning Models (LRRMs)*, has demonstrated that small recursive models can achieve strong performance on challenging reasoning benchmarks, including Sudoku-Extreme (Wang et al., 2025), ARC-AGI-1 (Chollet, 2019), and Maze-Hard (Lehnert et al., 2024), in some cases surpassing reasoning LLMs such as DeepSeek-R1, o3-mini-high, and Gemini 2.5 Pro with orders of magnitude fewer parameters. This is commonly attributed to the additional computational depth afforded by recursion: by reusing the same small set of parameters across many thinking steps, the model realizes an effective depth far beyond what its parameter count suggests. These results suggest that recursive computation and iterative refinement provide a promising alternative to conventional parameter scaling for structured reasoning.

While iterative computation can be trained via full unrolling (Dehghani et al., 2019) or truncated horizons (Geiping et al., 2026), models like HRM and TRM isolate per-step gradients by detaching recurrent states between thinking steps (Wang et al., 2025; Jolicoeur-Martineau, 2025). Crucially, while their gradient propagation is step-local, their supervision remains dense and uniform, applying the full target at every intermediate depth. For multi-step reasoning, this creates a fundamental mismatch: *early states lack the context to derive full targets, forcing premature guessing and injecting noisy gradients into shared parameters*, which we call ***state-supervision mismatch***. In Sudoku, for example, some cells may be directly determined from the current partial solution, whereas others become determined only after additional cells have been resolved. Applying the full target uniformly at this step therefore supervises both components that are compatible with the current reasoning state and components whose supporting information has not yet emerged.

Motivated by this observation, we propose **AS-TRM**, an **A**daptive **S**upervision strategy for **TRM** that aligns supervision with evolving reasoning progress. As illustrated in Figure 1, at each thinking

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

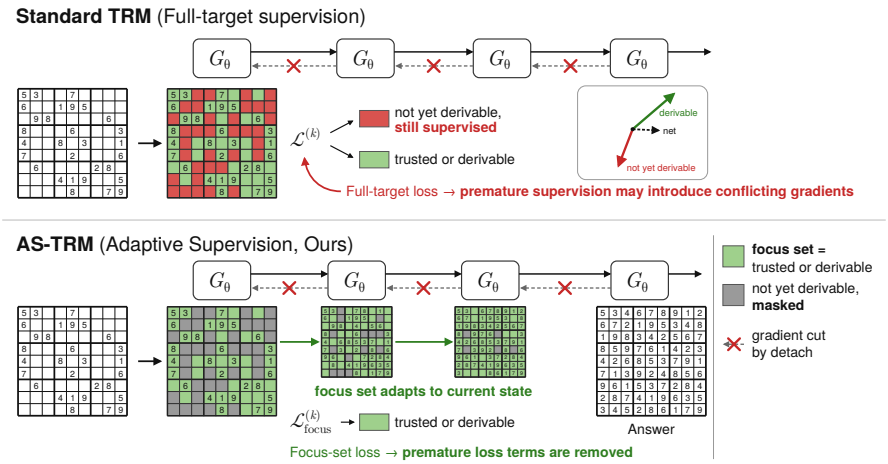


Figure 1: Overview of Standard TRM and AS-TRM. Top: Standard TRM supervises every cell against the full target after each thinking step k. The loss $\mathcal{L}^{(k)}$ therefore includes cells that are not yet derivable from the current state (red), whose gradients can conflict with those of trusted or derivable cells (green). Bottom: AS-TRM restricts the loss to a focus set of cells that are trusted or derivable at thinking step k (green) and masks the rest (gray). The focus set expands as reasoning progresses. In both methods, gradients are cut between thinking steps by detach (dashed arrows with a red cross). G_θ denotes the recursive reasoning module applied at each thinking step.

step we exploit fine-grained feedback to estimate which prediction targets are currently promising and which still require further reasoning on the model's current state. Rather than supervising all targets uniformly, AS-TRM restricts the loss at each step to a *focus set* of currently derivable targets and defers the remaining targets to later thinking steps. In this way, supervision evolves together with the recursive reasoning process, allowing each step to focus on learning targets that match its current reasoning capability.

In summary, our contributions are threefold:

- We identify a critical mismatch between the step-wise capacity and the complexity of the supervision target in TRMs: each intermediate loss backpropagates only through the immediately preceding thinking step, yet supervises the full targets. At early steps, states lack context to derive full targets, forcing premature guessing and injecting noisy gradients into shared parameters.
- We propose AS-TRM, a simple-yet-effective adaptive supervision method that tailors the supervised targets to the model's reasoning progress at each thinking step. AS-TRM improves final accuracy by up to 7.9 points on Sudoku-Extreme and 6.3 points on Maze-Hard over the strongest baselines, and speeds up training by up to $1.79\times$ on Sudoku-Extreme and $1.51\times$ on ARC-AGI-1.
- Our empirical analysis shows that AS-TRM stabilizes training, stimulates genuine reasoning, smooths loss landscape, and aligns better with the canonical learning objective.

We release our full code in the supplementary material for reproducibility.

2 Related Work

Latent Recursive Reasoning Models. LRRMs have drawn substantial interest for achieving strong performance with few parameters, particularly on Sudoku-Extreme (Wang et al., 2025) and ARC-AGI-1 (Chollet, 2019). Earlier designs, such as HRM (Wang et al., 2025), laid the foundation for latent recursive processing, which was subsequently refined by TRM (Jolicoeur-Martineau, 2025) to improve efficiency and reasoning capability. Recent extensions improve the efficiency of this recurrent computation from different angles. Equilibrium Reasoners (EqR; Huang et al., 2026) intervene on the input side of the recursion: randomized state initialization broadens which attractor basins are reachable during training, and noise injection perturbs the state fed into each iteration

to avoid premature convergence to incorrect fixed points, together shaping the dynamics so that additional test-time iterations reliably converge to solution-aligned attractors. Deep Improvement Supervision (DIS; Asadulaev et al., 2026) instead targets supervision. It casts recursive refinement as policy improvement and trains each step against an intermediate target obtained by corrupting the ground truth at a noise level fixed for that step. Because every target is tied to a step index, DIS runs a fixed number of steps and drops TRM's adaptive computation time (ACT; Graves, 2017). AS-TRM advances this direction by making supervision adaptive: it chooses which positions to supervise at each step from the model's current prediction, and it retains ACT, enabling dynamic allocation of thinking steps; see Appendix A.1 for extended discussion.

Adaptive Supervision and Curriculum Learning. Adaptive supervision changes what a model is supervised on during training, rather than using a fixed set of examples and targets. It can adapt which examples are trained on, or the targets themselves. Curriculum learning is a classic way to adapt examples: it presents them in a meaningful order, typically from easy to hard (Bengio et al., 2009). Self-paced learning derives this order from the learner, selecting examples by the model's current loss (Kumar et al., 2010). Dynamic oracles and DAgger instead adapt the target to the state reached by the model (Goldberg & Nivre, 2013; Ross et al., 2011). AS-TRM moves the easy-to-hard idea from training time to recursive depth, and, like these methods, makes the supervision state-dependent: at each thinking step, the supervised targets are determined by the state the model has reached, not by a fixed schedule or a step index. Unlike them, it does not supervise a complete oracle target: it supervises only the frontier of targets derivable from trusted units, and this frontier expands as recursion proceeds.

3 Background: Tiny Recursive Models

Latent recursive reasoning models (LRRMs) reuse a small set of parameters across multiple thinking steps to refine a latent state, and achieve strong performance on puzzle benchmarks such as Sudoku-Extreme, Maze-Hard, and ARC-AGI-1 with far fewer parameters than LLMs. TRM (Jolicoeur-Martineau, 2025) is a representative instance of LRRMs, whose formulation we describe below. We remark that AS-TRM is generally applicable to LRRMs, thgouh we evaluate it on TRMs due to their simple architecture design and competitive performance. Specifically, for a problem with n output positions $u \in \{1, \dots, n\}$ (e.g., the 81 cells of a Sudoku grid), TRM organizes iterative reasoning into two nested levels of recurrent computation across up to N_{sup} thinking steps (termed supervision steps in TRM). At the higher level, TRM progressively refines an answer prediction $y \in \mathbb{R}^{n \times d}$ and a latent scratchpad $z \in \mathbb{R}^{n \times d}$, where each position holds a d-dimensional embedding. At the lower level, a shared parameter set θ updates y and z at different frequencies governed by inner cycles (H, L) (Appendix B.1). Because AS-TRM leaves these inner dynamics and architecture untouched, we denote a single thinking step by the operator $G_\theta^{H,L}$:

$$(y^{(k)}, z^{(k)}) = G_\theta^{H,L}\Big(x,\, \text{sg}(y^{(k-1)}),\, \text{sg}(z^{(k-1)})\Big), \quad k = 1, \dots, N_{\text{sup}}, \tag{1}$$

where x is the problem input and $\text{sg}(\cdot)$ denotes the stop-gradient (detach) operator.

At each step k, TRM decodes the state $h^{(k)} := (y^{(k)}, z^{(k)})$ into an answer prediction $\hat{y}^{(k)} = f_O(y^{(k)})$ and computes a halting logit $q^{(k)} = f_Q(y^{(k)})$, which determines whether to halt or continue based on an adaptive computation time mechanism (Graves, 2017). Training applies supervision at every executed step:

$$\mathcal{L}^{(k)} = \ell(\hat{y}^{(k)}, y^\star) + \text{BCE}\Big(q^{(k)},\, \mathbf{1}[\arg\max \hat{y}^{(k)} = y^\star]\Big), \tag{2}$$

where $y^\star$ is the ground-truth solution, BCE is the binary cross-entropy on halting prediction, and $\ell(\hat{y}^{(k)}, y^\star) = \sum_{u=1}^{n} \ell_u^{(k)}$ sums the per-position cross-entropy terms $\ell_u^{(k)} = \text{CE}(\hat{y}_u^{(k)}, y_u^\star)$.

State-supervision mismatch. A fundamental training-inference gap arises from combining *step-local gradient updates* with *dense full-target supervision*. While inference requires continuous multi-step reasoning across the entire trajectory, step-local detachment (sg) isolates gradient updates to the current step k, preventing backpropagation through unrolled execution histories.[1]

[1]While one could backpropagate through the unrolled trajectory without stop-gradients, doing so underperforms the detached baseline in practice due to vanishing gradients and optimization instability (Appendix B.2).

Consequently, supervising the complete ground truth $y^\star$ at every intermediate step creates a structural mismatch with the model's current reasoning capacity. Intuitively, since ℓ decomposes additively across positions u, the target $y^\star$ can be partitioned at step k into positions that are currently derivable given state $(y^{(k-1)}, z^{(k-1)})$, and positions that remain non-derivable:

$$\ell(\hat{y}^{(k)}, y^\star) = \sum_{u \in \text{derivable}} \ell_u^{(k)} + \sum_{u \in \text{non-derivable}} \ell_u^{(k)}. \tag{3}$$

While the first term provides informative, stable gradients for positions supported by current context, the second term forces premature guessing on targets beyond the model's reasoning capacity at the current step. This forces shared parameters θ to fit targets that require future reasoning steps, injecting noisy gradients into updates (Figure 4); this is the key bottleneck that motivates our AS-TRM. AS-TRM formalizes this notion of derivability as part of the *focus set* introduced in Section 4, which restricts supervision to positions that are either already trusted or derivable from current state.

4 AS-TRM: Adaptive Supervision for Tiny Recursive Models

Motivated by the analysis and intuition in Section 3, in this section, we propose AS-TRM, an adaptive supervision algorithm that dynamically selects which part of the answer should be supervised, tailored to the current reasoning state. Note that AS-TRM is a plug-and-play module that requires minimal changes to model architecture or training regime.

4.1 State-Conditioned Adaptive Supervision over Recursive Depth

At intermediate thinking steps, the model need not be equally ready to predict every part of the final solution. Some parts are already supported by the current state, some lie just beyond what has been resolved, and others may still require additional computation. Applying the full target at every step ignores this asymmetry. Therefore, we propose the following:

> **AS-TRM Guiding Principle.** Place supervision near the model's *current reasoning frontier* to preserve what is already supported, extend supervision to a bounded next frontier, and defer targets that require further computation.

At training iteration t and thinking step k, the model produces a prediction $\hat{y}_t^{(k)}$, whose coordinates we call *positions* (e.g., the cells of a Sudoku grid or of a maze). We partition the positions into a *focus set* $\mathrm{F}_t^{(k)}$ and a *non-focus set* $\overline{\mathrm{F}}_t^{(k)}$. The focus set contains the positions that are currently appropriate for supervision: those that are already trusted, together with a bounded next frontier that can be derived and reached from the current state. Abstractly,

$$\mathrm{F}_t^{(k)} = \mathrm{T}_t^{(k)} \cup \mathrm{R}_t^{(k)}, \tag{4}$$

where $\mathrm{T}_t^{(k)}$ denotes the currently trusted part of the prediction and $\mathrm{R}_t^{(k)}$ denotes a derivable frontier. The non-focus set $\overline{\mathrm{F}}_t^{(k)}$ contains the remaining output positions.

Crucially, $\mathrm{F}_t^{(k)}$ is recomputed from the current prediction after every thinking step. The membership can therefore evolve with computational depth: a target position that is not selected for supervision at the current step may become eligible after further recursive computation. Likewise, as training progresses and the model improves, a position that is not yet eligible at iteration t may become eligible at a later iteration. For example, there may exist a target position u, some $r > 0$, and some $\Delta t > 0$ such that

$$u \in \overline{\mathrm{F}}_t^{(k)} \qquad \text{but} \qquad u \in \mathrm{F}_t^{(k+r)} \quad \text{or} \quad u \in \mathrm{F}_{t+\Delta t}^{(k)}.$$

Thus, the effective supervision region tends to expand along both *recursive depth* and *training progress*.

The construction of $\mathrm{T}_t^{(k)}$ and $\mathrm{R}_t^{(k)}$ follows the natural granularity of reasoning in each task. For Sudoku and Maze, progress is locally decomposable: individual cells can become reliable or derivable before the full solution is complete, so we construct the focus set at the cell level. In Sudoku,

Table 1: Task-specific construction of the focus set. Full selection rules and parameter settings are given in Table 7 (Appendix C.2).

Task	Unit	Trusted set $\mathrm{T}_t^{(k)}$	Frontier $\mathrm{R}_t^{(k)}$	λ
Sudoku-Extreme	Cell	Correct, high-confidence cells	Cells derivable by constraint propagation	0
Maze-Hard	Cell	High-confidence shortest-path cells predicted as path	Shortest-path cells within $0.2d^\star$ layers of the trusted chains from both ends	0
ARC-AGI-1	Puzzle	Puzzles predicted entirely correctly	Unsolved puzzles with the lowest loss	0.11

the frontier is defined through local constraint propagation, while in Maze it follows the currently supported shortest-path frontier, taken along the shortest path most consistent with the current prediction rather than the single dataset label. In ARC-AGI-1, progress is less naturally decomposed over individual output cells because the prediction is governed by a global transformation; we therefore construct the focus set at the puzzle level, i.e., $\mathrm{F}_t^{(k)}$ then contains all positions of the selected puzzles. Table 1 summarizes the corresponding definitions of $\mathrm{T}_t^{(k)}$ and $\mathrm{R}_t^{(k)}$ and their main settings; full selection rules and parameter settings are given in Appendix C.2.

4.2 Adaptive Supervision Objective

Recall $\ell_u^{(k)}$, the per-position answer loss at output position u after thinking step k (Section 3), and the focus set $\mathrm{F}_t^{(k)}$ from Equation (4); below we write $\mathrm{F}^{(k)}$ and $\overline{\mathrm{F}}^{(k)}$, dropping the training-iteration index t when it is clear from context. AS-TRM assigns unit weight to focus positions and relative weight $\lambda \in [0, 1]$ to non-focus positions:

$$\mathcal{L}_{\text{AS-TRM}}^{(k)} = \frac{\sum_{u \in \mathrm{F}^{(k)}} \ell_u^{(k)} + \lambda \sum_{u \in \overline{\mathrm{F}}^{(k)}} \ell_u^{(k)}}{|\mathrm{F}^{(k)}| + \lambda |\overline{\mathrm{F}}^{(k)}|}. \tag{5}$$

Thus $\lambda = 0$ completely defers non-focus supervision, while $\lambda = 1$ recovers dense answer supervision (for scale-invariant optimizers such as Adam). The standard TRM halting loss is added unchanged. In experiments, we use λ close to or equal to 0.

At each thinking step k, AS-TRM therefore (i) runs TRM computation $G_\theta^{H,L}$, (ii) constructs $\mathrm{F}^{(k)}$ from the detached current prediction using the rule in Table 1, (iii) applies Equation (5) to the answer loss together with the unchanged halting loss, and (iv) detaches the recurrent states before step $k+1$.

5 Experimental Evaluation

In this section, we evaluate AS-TRM on three puzzle-like tasks that involve heavy reasoning: Sudoku-Extreme (Wang et al., 2025), Maze-Hard (Wang et al., 2025; Lehnert et al., 2024), and ARC-AGI-1 (Chollet, 2019). Most setups and evaluation protocols follow the literature (Jolicoeur-Martineau, 2025; Wang et al., 2025). We first introduce the setup and tasks, then present the main results, and finally report ablations and analysis.

5.1 Experimental Setup

Training. All methods use TRM as the backbone (Section 3) with up to $N_{\text{sup}} = 16$ thinking steps; AS-TRM only changes which output positions are supervised. Hyperparameters follow TRM and, together with the method-specific settings of the baselines, are listed in Appendices C.3 and C.4. We report the mean and standard deviation over multiple seeds (Appendix C.4).

Tasks. Sudoku-Extreme (Wang et al., 2025) has 1,000 training puzzles, each with 1,000 augmented variants, and 422,786 test puzzles. Maze-Hard (Wang et al., 2025; Lehnert et al., 2024) has 1,000 training and 1,000 evaluation 30×30 mazes, each admitting many shortest paths of length at least 110, though the dataset labels only one (Appendix C.1). We therefore report valid shortest path accuracy rather than exact match against the label. Following TRM, on ARC-AGI-1 (Chollet, 2019) we train on the 400 training tasks, 160 ConceptARC tasks (Moskvichev et al., 2023), and the demonstration pairs of the 400 evaluation tasks. We report pass@2 on the evaluation test inputs and compare all methods at around 120k steps (Table 11).

Baselines. We compare against self-paced learning (SPL) (Kumar et al., 2010), which trains on the examples the model currently finds easiest, and Deep Improvement Supervision (DIS) (Asadulaev et al., 2026), which supervises each thinking step with a scheduled intermediate target, runs a fixed $N_{\text{sup}} = 6$ steps as in its paper, and drops the halting head; details can be found in Appendix C.3.

Evaluation. Prior work reports @max, which always runs all N_{sup} thinking steps. This ignores the learned halting head, and because TRM can overwrite an already-correct answer, as observed for HRM by Ren & Liu (2026), @max fluctuates strongly during training. We therefore report @halt-or-max, which stops at the first step where the halting head fires or at the step budget, for both Table 2 and Figure 2. We measure speedup as $S_{\text{TRM}}(a)/S_{\text{AS-TRM}}(a)$, where $S_m(a)$ is the first training step at which the seed-averaged curve of method m reaches accuracy a; we report the largest value over a. We also report the @max results at Appendix C.5.4 since some of the results are carried from references, which by design use @max.

5.2 Main Results

Overall, AS-TRM improves both the final performance (Table 2) and training efficiency (Figure 2) of TRM. We additionally compare against self-paced learning (Kumar et al., 2010) to test whether these improvements can be attributed to a generic easy-to-hard curriculum. The results of TRM + SPL show that directly applying self-paced learning to TRM does not consistently improve the original TRM baseline, indicating that generic sample-level scheduling is insufficient; and that our state-conditioned adaptive supervision that expands the frontier at a finer granularity substantially improves performance. AS-TRM also outperforms DIS (Asadulaev et al., 2026), which removes the halting head and always runs a fixed number of steps, on Sudoku-Extreme and Maze-Hard, and is on par with it on ARC-AGI-1 (Table 2). Rather than discarding adaptive halting, AS-TRM makes better use of it: the halting step grows with puzzle difficulty on Sudoku-Extreme while exact match stays higher (Appendix C.6.5).

As shown in Table 2, AS-TRM improves TRM on Sudoku-Extreme at all three recursion depths and on Maze-Hard. On Sudoku-Extreme, it improves exact match by $+3.7$, $+7.9$, and $+6.6$ points across the three compute settings, and on Maze-Hard, AS-TRM combined with flip and rotation augmentation (Appendix C.5.3) improves valid shortest path accuracy by $+6.3$ points over the best baseline under the standard TRM training setup. In contrast, self-paced learning does not consistently improve over TRM, suggesting that the gains are not explained by generic easy-to-hard scheduling. Figure 2 presents the training curve across the three tasks. It demonstrates that AS-TRM brings consistent speedups, especially in early training stages. The speed-up rate is up to $1.79\times$, $1.53\times$, and $1.51\times$ on Sudoku-Extreme, Maze-Hard, and ARC-AGI-1, respectively.

5.3 Ablations and Analysis

We conduct systematic ablation and sensitivity studies (Appendices C.5.1 and C.5.2), which show that the choice of the focus set matters: on Sudoku-Extreme, AS-TRM outperforms both a random and an inverted choice of the same size. AS-TRM's the gain are insensitive to the trust threshold above a certain threshold, and they shrink as non-focus targets receive more weight. We then analyze how adaptive supervision changes learning along three lines: the recursion it produces, the loss landscape, and the alignment of its gradients with the unrolled terminal gradient.

5.3.1 Adaptive Supervision Stimulates Genuine Reasoning

We find that adaptive supervision yields a recursion that keeps a correct answer and continues to improve with more iterations, whereas the recursion of TRM overwrites it. Figure 3 traces the Sudoku-Extreme gain from the metric to the recursion dynamics, all at $(H, L) = (2, 4)$. Solving a puzzle requires the model to both finish reasoning (i.e., halt) within budget and exactly match the solution when halting. As a result, we decompose these two factors and name them coverage@halt and exact match@halt:

$$\underbrace{\Pr[\text{solve}]}_{\text{exact match @halt-or-max}} = \underbrace{\Pr[\text{halt within budget}]}_{\text{coverage@halt}} \cdot \underbrace{\Pr[\text{solve} \mid \text{halt within budget}]}_{\text{exact match@halt}},$$

Table 2: Main results under @halt-or-max evaluation. Evaluation setup and analyses are discussed in Sections 5.1 and 5.2.

	Sudoku-Extreme			Maze-Hard	ARC-AGI-1
	(exact match)			(valid shortest path)	(pass@2)
D	8 / 6	24 / 20	48 / 42	8 / 6	24 / 30[2]
(H, L), HRM (27M)	(1,1)	(2,2)	(3,3)	(1,1)	(2,2)[2]
(H, L), others (7M)	(1,2)	(2,4)	(3,6)	(1,2)	(3,4)
Direct Pred	0.0*	0.0*	0.0*	0.0*	21.0*
HRM	46.4*	55.0*	61.6*	81.0 (4.9)	40.3*
TRM	<u>59.9</u> (0.7)	<u>69.7</u> (1.3)	<u>71.4</u> (2.2)	86.8 (2.7)	40.7 (1.2)[1]
TRM + SPL	59.5 (1.5)	65.4 (1.7)	69.5 (4.5)	<u>86.9</u> (1.5)	38.1 (1.8)
DIS[3]	38.0 (1.1)	64.8 (2.5)	69.2 (1.6)	72.0 (2.2)	<u>41.3</u>*
AS-TRM	**63.6** (0.9)	**77.6** (1.8)	**78.0** (2.0)	**93.2** (1.9)	**41.4** (1.5)
Δ	+3.7	+7.9	+6.6	+6.3	+0.1

Note. Δ is the improvement of AS-TRM over best existing method in each column. Best result per column in bold, second best underlined. Small numbers in parentheses are standard deviations across seeds. Due to late-training decline for both TRM and AS-TRM (Figure 2), we report the peak value for all methods. $D = H(L+1)\, n_{\text{layers}}$ is the approximate effective depth per thinking step. HRM uses two 4-layer networks and all other methods a single 2-layer network, so a comparable D is reached at different (H, L). Parameter counts are given in the row labels. Direct Pred is a non-recursive Transformer of comparable size trained to predict the answer directly, as reported by Wang et al. (2025).

[1] Our reproduction, slightly lower than reported in Jolicoeur-Martineau (2025). AS-TRM trains up to $1.51\times$ faster in the early stage (Figure 2).

[2] On ARC-AGI-1 we follow the released configuration from Jolicoeur-Martineau (2025), $(H, L) = (3, 4)$ with $D = 30$. HRM is quoted at its own reported setting $(2, 2)$ with $D = 24$, so this column is not depth-matched.

[3] Deep Improvement Supervision (Asadulaev et al., 2026), our reproduction on Sudoku-Extreme and Maze-Hard; details in Appendix C.3.2.

* Taken from Wang et al. (2025); Jolicoeur-Martineau (2025); Asadulaev et al. (2026) under their reported evaluation protocols and included for reference.

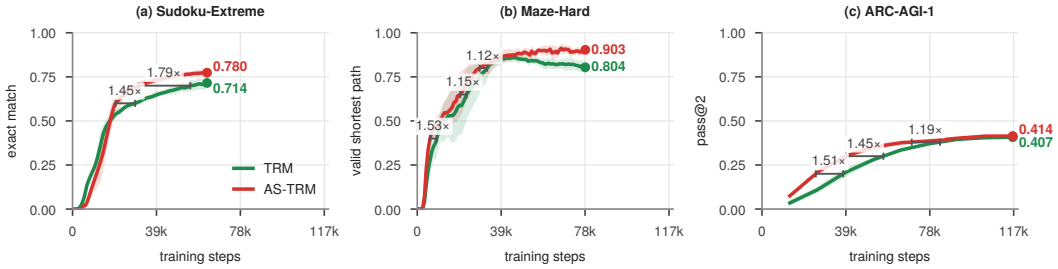


Figure 2: Training curves under @halt-or-max. Sudoku-Extreme uses $(H, L) = (3, 6)$. Labels at the end of each curve are final-step values, which can be lower than the peak values in Table 2. The speed-up rate is up to $1.79\times$, $1.53\times$, and $1.51\times$ on Sudoku-Extreme, Maze-Hard, and ARC-AGI-1.

In subfigure (a), we find that exact match@halt is very high for both methods (slope 0.989), so the improvement comes from coverage@halt, the fraction of puzzles that halt within budget: AS-TRM halts on more puzzles, and sooner (subfigure (b)).

We then disable halting and roll the recursion out for up to 1024 steps, far beyond the training budget of 16 (subfigures (c) and (d)). Following Equilibrium Reasoners (EqR; Huang et al., 2026), we track the relative residual of the answer state, $\delta_k = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \|y_u^{(k)} - y_u^{(k-1)}\| / \|y_u^{(k-1)}\|$, where $\mathcal{U}$ is the set of non-clue cells of a puzzle. TRM solves puzzles and then revises them away: its exact match peaks early and then falls, and its residual does not decrease. In contrast, the residual of AS-TRM keeps decreasing until more than half of the puzzles stop changing, and its exact match keeps improving past the training budget, from 0.78 at step 16 to 0.95 at step 1024. Halting also adapts to

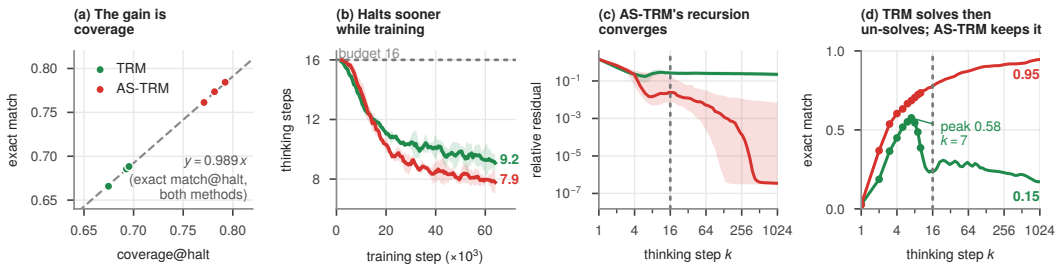


Figure 3: (a) Exact match@halt-or-max against coverage@halt; dashed line $y = 0.989x$. (b) Thinking steps per training puzzle, mean $\pm$ sd over runs. (c, d) Rollout without halting on 512 Sudoku-Extreme test puzzles, far beyond the training budget of 16 steps (dotted line): relative residual of the answer state (median and interquartile range over puzzles) and exact match. The horizontal axis is logarithmic, with unlabeled ticks at the powers of two in between. Setup in Appendix C.6.4.

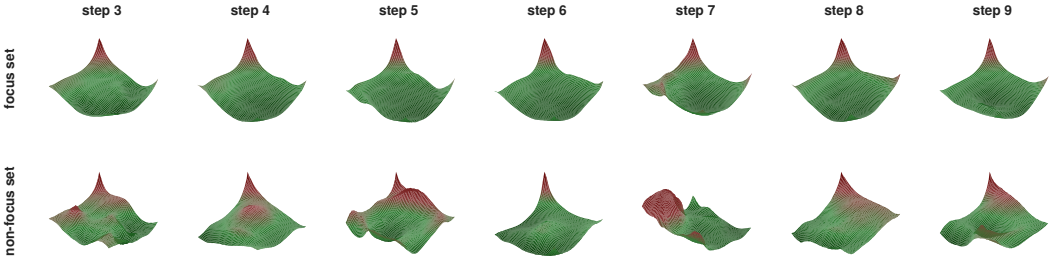


Figure 4: Loss landscape of a representative Sudoku-Extreme puzzle across thinking steps. Details in Appendix C.6.2.

difficulty: harder puzzles get more thinking steps and AS-TRM is more accurate than TRM at every difficulty tier of the end-of-training checkpoints analyzed there (Figure 16, Appendix C.6.5).

5.3.2 Adaptive Supervision Smooths the Loss Landscape

Following Li et al. (2018), we visualize the loss landscape. Figure 4 shows that the focus-set loss landscape remains visually similar across thinking steps, while the non-focus landscape changes substantially (curvature flip rate 0.39 vs. 0.03 for the focus loss; details in Appendix C.6.2), indicating that adaptive supervision provides a smoother and more consistent loss landscape across steps and filters out rugged loss components.

5.3.3 Gradients Align with the Unrolled Terminal Gradient

Canonically, the TRM training should minimize the negative log-likelihood of the ground-truth solution after multiple training steps N_{sup}, which implies the loss $\ell^{(N_{\text{sup}})}$. An unrolled training would thus lead to the contribution of the gradient from step k in the form of $J^{(k)\top}\,\Phi^{(N_{\text{sup}},k)\top}\,\nabla\ell^{(N_{\text{sup}})}$, where $J^{(k)} = \frac{\partial h^{(k)}}{\partial \theta}$ the Jacobian of the state after step k with respect to the parameters θ, and $\Phi^{(N_{\text{sup}},k)} = \frac{\partial h^{(N_{\text{sup}})}}{\partial h^{(k)}}$ is the Jacobian of the final state with respect to state after step k. Conditioned on correct positions $\mathrm{C}^{(N_{\text{sup}})}$ at step N_{sup}, we thus call this step k gradient contribution the *unrolled terminal gradient*:

$$g^{(k)}_{\text{term}} = J^{(k)\top}\,\Phi^{(N_{\text{sup}},k)\top}\,\nabla\ell^{(N_{\text{sup}})}_{\mathrm{C}^{(N_{\text{sup}})}}, \tag{6}$$

Though direct unrolled training leads to poor performance due to optimization instability (Appendix B.2), we view this unrolled terminal gradient direction $g^{(k)}_{\text{term}}$ an important reference direction since it reflects the inference-aligned direction that improves correctness likelihood.

On the other hand, TRM detaches the state between thinking steps, so step k is trained only through its own log-likelihood-based loss $\ell^{(k)}$. We hypothesize that restricting $\ell^{(k)}$ to the focus set gives a local gradient that is a better proxy for this missing signal: a focus position is likely to keep its value

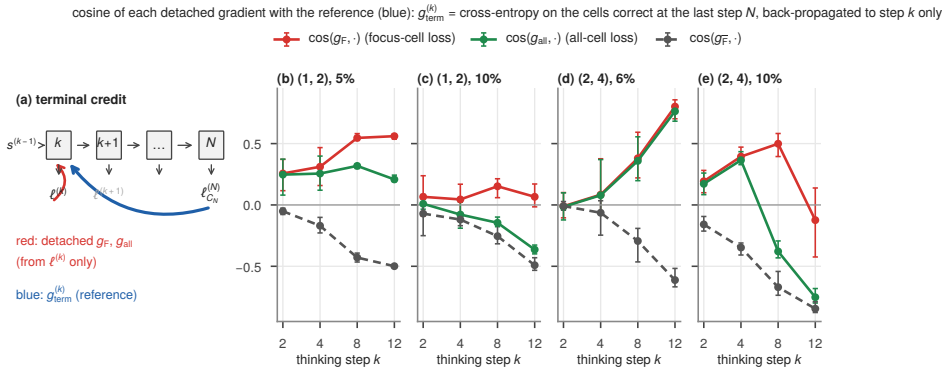


Figure 5: (a) Illustration of the detached gradients and the terminal gradient $g^{(k)}_{\text{term}}$. (b) to (e) Cosine similarity of $g^{(k)}_{\text{F}}$, $g^{(k)}_{\text{all}}$, and $g^{(k)}_{\overline{\text{F}}}$ with $g^{(k)}_{\text{term}}$ at each thinking step k. Panel titles give (H, L) and the training progress. Error bars show min and max over three batches of 768 puzzles each. Finding: detached focus-set gradient aligns with terminal gradient, while non-focus set gradient diverges.

in later steps, so its gradient at step k resembles its final gradient, whereas a non-focus position is still likely to change.

Let $\ell^{(k)}_{\text{F}} = \sum_{u \in \text{F}^{(k)}} \ell^{(k)}_u$ and $\ell^{(k)}_{\overline{\text{F}}} = \sum_{u \in \overline{\text{F}}^{(k)}} \ell^{(k)}_u$ be the unnormalized losses on the focus and non-focus positions, The detached gradients are

$$g^{(k)}_{\text{all}} = J^{(k)\top} \nabla \ell^{(k)}, \qquad g^{(k)}_{\text{F}} = J^{(k)\top} \nabla \ell^{(k)}_{\text{F}}, \qquad g^{(k)}_{\overline{\text{F}}} = g^{(k)}_{\text{all}} - g^{(k)}_{\text{F}}. \tag{7}$$

To verify the hypothesis, we compute the cosine similarity between $g^{(k)}_{\text{term}}$ and each of $g^{(k)}_{\text{F}}$, $g^{(k)}_{\text{all}}$, and $g^{(k)}_{\overline{\text{F}}}$ on the same TRM checkpoint. In nearly all settings (Figure 5), $g^{(k)}_{\text{F}}$ is more aligned with $g^{(k)}_{\text{term}}$ than $g^{(k)}_{\text{all}}$, whereas $g^{(k)}_{\overline{\text{F}}}$ is anti-aligned with it; this holds only early in training (Figure 17), since the norm of $g^{(k)}_{\text{term}}$ soon grows by several orders of magnitude above the detached gradient and all cosines approach zero. The focus and non-focus gradients also conflict directly, with a negative inner product for all pairs of thinking steps (Figure 12 in Appendix C.6.1).

6 Discussion

Our results suggest that adaptive supervision can operate within a single example along recursive depth. As the model's state changes, so does the set of output components that can provide useful supervision. This connects AS-TRM to active or online selection of supervision: a training policy can decide which parts of an example should exert a gradient at each thinking step.

The same principle may apply more broadly across LRRMs, as well as to looped transformers (Saunshi et al., 2025; Geiping et al., 2026; Jeddi et al., 2026) and recursive language models (Zhang et al., 2025), when intermediate predictions are supervised through a limited gradient horizon. The central challenge is obtaining a reliable signal of partial progress. Task structure provides such a signal for Sudoku and Maze, but many domains lack a readily verifiable frontier. In these domains, confidence or self-generated intermediate answers may be frontier proxy. We think the success of AS-TRM delivers promising active supervision avenues for these broader domains and architectures, and we leave further investigation in these scenarios as future work.

7 Conclusion

We identified a state-supervision mismatch along recursive depth in LRRMs trained with deep supervision and step-local gradients, such as TRMs, where each thinking step is supervised with the full answer although it can derive only part of it, and proposed AS-TRM, which adaptively selects the supervised targets from the model's current predictions, so that supervision expands with the model's reasoning progress along both recursive depth and training. Instantiated on TRM, this simple change drastically improves training effectiveness or training efficiency on all three tasks. We hope this work provides insight into training LRRMs more efficiently and effectively.

AI USE STATEMENT

We used AI tools for the following tasks:

- Propose or refine hypotheses: To strengthen the paper, we asked agents to suggest additional diagnostics.
- Design or provide feedback on research methodology or experiments: Coding agents were asked to provide feedback throughout the project, for example on the choice of the terminal reference gradient and of the control settings in the gradient analysis. They also drew visualizations that helped us diagnose the training and the recursion dynamics.
- Implement methods: Coding agents implemented the analysis and diagnostic code (gradient alignment with the unrolled terminal gradient, long rollouts without halting, attention and residual analyses) and the unrolled (no stop-gradient) experiment.
- Support qualitative and thematic data analysis: We often asked coding agents to create tables and figures of results so we could interpret the results. Many of these figures and tables ended up in the final draft.

Additional disclosures include the use of AI for the following purposes: create or modify scientific figures or images, create or edit software code, debugging code, visualizing error cases, brainstorming, and edit a research paper to improve readability, including polishing and shortening the text, checking typos and notation consistency, and paper formatting (LaTeX polishing).

On the other hand, the core research idea, algorithm design, experiment design, implementation, and experiment execution still largely stem from manual efforts. The overall role of AI tools here is still auxiliary rather than primary. We have reviewed and validated all AI-assisted work and take responsibility for the final content of the work, including text, claims, and artifacts produced with the aid of AI.

ETHICS STATEMENT

This work studies training methods for LRRMs on algorithmic tasks. All experiments use synthetic puzzles or publicly available benchmarks. No human subjects, personal data, or sensitive content are involved. We do not foresee direct negative societal impacts beyond those generally associated with advances in machine learning. We have read and adhere to the ICLR Code of Ethics.

REPRODUCIBILITY STATEMENT

We describe our method in Section 4 and the experimental setup in Section 5. Full hyperparameters, model configurations, data preprocessing, and training details are provided in Appendix C.4. We include anonymized source code in the supplementary material to reproduce our main results.

REFERENCES

Mohammad Hossein Amani, Aryo Lotfi, Nicolas Mario Baldwin, Samy Bengio, Mehrdad Farajtabar, Emmanuel Abbe, and Robert West. Rl for reasoning by adaptively revealing rationales, 2026. URL https://arxiv.org/abs/2506.18110.

Arip Asadulaev, Rayan Banerjee, Fakhri Karray, and Martin Takac. Latent reasoning in trms is secretly a policy improvement operator, 2026. URL https://arxiv.org/abs/2511.16886.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pp. 41–48, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605585161. doi: 10.1145/1553374.1553380. URL https://doi.org/10.1145/1553374.1553380.

François Chollet. On the measure of intelligence, 2019. URL https://arxiv.org/abs/1911.01547.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers, 2019. URL https://arxiv.org/abs/1807.03819.

Katie Everett, Lechao Xiao, Mitchell Wortsman, Alexander A. Alemi, Roman Novak, Peter J. Liu, Izzeddin Gur, Jascha Sohl-Dickstein, Leslie Pack Kaelbling, Jaehoon Lee, and Jeffrey Pennington. Scaling exponents across parameterizations and optimizers, 2024. URL https://arxiv.org/abs/2407.05872.

Richard Freinschlag, Timo Bertram, Erich Kobler, Andreas Mayr, and Günter Klambauer. Symbol-equivariant recurrent reasoning models, 2026. URL https://arxiv.org/abs/2603.02193.

Zitian Gao, Lynx Chen, Yihao Xiao, He Xing, Ran Tao, Haoming Luo, Joey Zhou, and Bryan Dai. Universal reasoning model, 2025. URL https://arxiv.org/abs/2512.14693.

Jonas Geiping, Sean McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. Scaling up test-time compute with latent reasoning: A recurrent depth approach. *Advances in Neural Information Processing Systems*, 38: 41340–41391, 2026.

Yoav Goldberg and Joakim Nivre. Training deterministic parsers with non-deterministic oracles. *Transactions of the Association for Computational Linguistics*, 1:403–414, 2013. doi: 10.1162/tacl_a_00237. URL https://aclanthology.org/Q13-1033/.

Alex Graves. Adaptive computation time for recurrent neural networks, 2017. URL https://arxiv.org/abs/1603.08983.

Guy Hacohen, Avihu Dekel, and Daphna Weinshall. Active learning on a budget: Opposite strategies suit high and low budgets, 2022. URL https://arxiv.org/abs/2202.02794.

Benhao Huang, Zhengyang Geng, and Zico Kolter. Equilibrium reasoners: Learning attractors enables scalable reasoning, 2026. URL https://arxiv.org/abs/2605.21488.

Ahmadreza Jeddi, Marco Ciccone, and Babak Taati. Loopformer: Elastic-depth looped transformers for latent reasoning via shortcut modulation. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=RzYXb5YWBs.

Alexia Jolicoeur-Martineau. Less is more: Recursive reasoning with tiny networks, 2025. URL https://arxiv.org/abs/2510.04871.

Siddharth Karamcheti, Ranjay Krishna, Li Fei-Fei, and Christopher D. Manning. Mind your outliers! investigating the negative impact of outliers on active learning for visual question answering, 2021. URL https://arxiv.org/abs/2107.02331.

M. Pawan Kumar, Benjamin Packer, and Daphne Koller. Self-paced learning for latent variable models. In *Proceedings of the 24th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'10, pp. 1189–1197, Red Hook, NY, USA, 2010. Curran Associates Inc.

Lucas Lehnert, Sainbayar Sukhbaatar, DiJia Su, Qinqing Zheng, Paul Mcvay, Michael Rabbat, and Yuandong Tian. Beyond a*: Better planning with transformers via search dynamics bootstrapping, 2024. URL https://arxiv.org/abs/2402.14083.

David D. Lewis and William A. Gale. A sequential algorithm for training text classifiers, 1994. URL https://arxiv.org/abs/cmp-lg/9407020.

Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets, 2018. URL https://arxiv.org/abs/1712.09913.

Zhiyuan Li, Hong Liu, Denny Zhou, and Tengyu Ma. Chain of thought empowers transformers to solve inherently serial problems, 2024. URL https://arxiv.org/abs/2402.12875.

Zhenghao Lin, Zhibin Gou, Yeyun Gong, Xiao Liu, Yelong Shen, Ruochen Xu, Chen Lin, Yujiu Yang, Jian Jiao, Nan Duan, and Weizhu Chen. Rho-1: Not all tokens are what you need, 2025. URL https://arxiv.org/abs/2404.07965.

William Merrill and Ashish Sabharwal. A little depth goes a long way: The expressive power of log-depth transformers, 2025. URL https://arxiv.org/abs/2503.03961.

Sören Mindermann, Jan Brauner, Muhammed Razzak, Mrinank Sharma, Andreas Kirsch, Winnie Xu, Benedikt Höltgen, Aidan N. Gomez, Adrien Morisot, Sebastian Farquhar, and Yarin Gal. Prioritized training on points that are learnable, worth learning, and not yet learnt, 2022. URL https://arxiv.org/abs/2206.07137.

Arseny Moskvichev, Victor Vikram Odouard, and Melanie Mitchell. The conceptarc benchmark: Evaluating understanding and generalization in the arc domain, 2023. URL https://arxiv.org/abs/2305.07141.

Zirui Ren and Ziming Liu. Are your reasoning models reasoning or guessing? a mechanistic analysis of hierarchical reasoning models, 2026. URL https://arxiv.org/abs/2601.10679.

Stephane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning, 2011. URL https://arxiv.org/abs/1011.0686.

Nikunj Saunshi, Nishanth Dikkala, Zhiyuan Li, Sanjiv Kumar, and Sashank J Reddi. Reasoning with latent thoughts: On the power of looped transformers. In *International Conference on Learning Representations*, volume 2025, pp. 14855–14881, 2025.

Burr Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009.

Swabha Swayamdipta, Roy Schwartz, Nicholas Lourie, Yizhong Wang, Hannaneh Hajishirzi, Noah A. Smith, and Yejin Choi. Dataset cartography: Mapping and diagnosing datasets with training dynamics, 2020. URL https://arxiv.org/abs/2009.10795.

Guan Wang, Jin Li, Yuhao Sun, Xing Chen, Changling Liu, Yue Wu, Meng Lu, Sen Song, and Yasin Abbasi Yadkori. Hierarchical reasoning model, 2025. URL https://arxiv.org/abs/2506.21734.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023. URL https://arxiv.org/abs/2201.11903.

Alex L Zhang, Tim Kraska, and Omar Khattab. Recursive language models. *arXiv preprint arXiv:2512.24601*, 2025.

Rui-Jie Zhu, Zixuan Wang, Kai Hua, Tianyu Zhang, Ziniu Li, Haoran Que, Boyi Wei, Zixin Wen, Fan Yin, He Xing, Lu Li, Jiajun Shi, Kaijing Ma, Shanda Li, Taylor Kergan, Andrew Smith, Xingwei Qu, Mude Hui, Bohong Wu, Qiyang Min, Hongzhi Huang, Xun Zhou, Wei Ye, Jiaheng Liu, Jian Yang, Yunfeng Shi, Chenghua Lin, Enduo Zhao, Tianle Cai, Ge Zhang, Wenhao Huang, Yoshua Bengio, and Jasofn Eshraghian. Scaling latent reasoning via looped language models, 2026. URL https://arxiv.org/abs/2510.25741.

Appendix Contents

A **Extended Related Work** . . . 15
A.1 Connection to Latent Recursive Reasoning Models . . . 15
A.2 Connection to Looped Models . . . 15
A.3 Connection to Curriculum Learning . . . 15
A.4 Connection to Active Learning . . . 16
A.5 Connection to Closely Related Work . . . 16
B **Latent Recursive Reasoning Models (LRRM)** . . . 18
B.1 Architecture Comparison between HRM and TRM . . . 18
B.2 Unrolled Experiment: Removing the Stop-Gradient . . . 18
C **Experimental Details** . . . 20
C.1 Task Examples . . . 20
C.1.1 Sudoku-Extreme . . . 20
C.1.2 Maze-Hard . . . 20
C.1.3 ARC-AGI-1 . . . 21
C.2 AS-TRM: Focus-Set Construction . . . 22
C.2.1 Sudoku-Extreme . . . 22
C.2.2 Maze-Hard . . . 22
C.2.3 ARC-AGI-1 . . . 22
C.3 Baselines . . . 23
C.3.1 Self-Paced Learning (SPL) . . . 23
C.3.2 Deep Improvement Supervision (DIS) . . . 23
C.4 Hyperparameters and Compute . . . 24
C.5 Additional Ablations . . . 25
C.5.1 Sudoku-Extreme Frontier Ablation . . . 25
C.5.2 Sensitivity Analysis on Sudoku-Extreme . . . 25
C.5.3 Maze-Hard Data Augmentation . . . 25
C.5.4 Results under @max Evaluation . . . 26
C.6 Details for Experimental Analysis . . . 27
C.6.1 Gradient Interference . . . 27
C.6.2 Loss Landscape . . . 27
C.6.3 Attention Dynamics on a Single Puzzle . . . 28
C.6.4 Convergence of the Recursion . . . 28
C.6.5 Halting Adapts to Puzzle Difficulty . . . 29
C.6.6 Alignment with the Terminal Credit . . . 30
D **Failed Experiments** . . . 33

Table 3: Notation used throughout the paper.

Symbol	Meaning
Model and recursion	
x	problem input
n, $u \in \{1, \dots, n\}$	number of output positions (e.g. the cells of a grid) and a position index
$y, z \in \mathbb{R}^{n \times d}$	answer state and latent scratchpad, d-dimensional embedding per position
$h^{(k)}$	state $(y^{(k)}, z^{(k)})$ after step k
(H, L)	numbers of high-level and low-level inner cycles
θ	parameters shared across all thinking steps
$G_\theta^{H,L}$	one thinking step, $h^{(k)} = G_\theta^{H,L}(x, \mathrm{sg}(h^{(k-1)}))$
$\mathrm{sg}(\cdot)$	stop-gradient (detach)
$k \in \{1, \dots, N_{\mathrm{sup}}\}$	thinking step (supervision step); N_{sup} is the maximum number of steps
$\hat{y}^{(k)}$, $q^{(k)}$	answer prediction $f_O(y^{(k)})$ and halting logit $f_Q(y^{(k)})$ after step k
f_O, f_Q	answer decoder and halting head
t	training iteration (optimizer step)
n_{layers}	number of layers of the recurrent network
D	effective depth per thinking step, $H(L+1)\, n_{\mathrm{layers}}$
Supervision	
$y^\star$	ground-truth solution
$\ell_u^{(k)}$, $\ell^{(k)}$	cross-entropy at position u after step k, and its sum over positions
CE, BCE, $\mathbf{1}[\cdot]$	cross-entropy, binary cross-entropy, indicator
$\mathcal{L}^{(k)}$	loss of step k in TRM: answer loss plus halting BCE
$\mathcal{L}^{(k)}_{\text{AS-TRM}}$	adaptive supervision loss of step k (Equation (5))
$\mathrm{F}_t^{(k)}$, $\overline{\mathrm{F}}_t^{(k)}$	focus and non-focus set of positions (indices t, k dropped when clear)
$\mathrm{T}_t^{(k)}$, $\mathrm{R}_t^{(k)}$	trusted set and derivable frontier, $\mathrm{F} = \mathrm{T} \cup \mathrm{R}$
$\ell_{\mathrm{F}}^{(k)}$, $\ell_{\overline{\mathrm{F}}}^{(k)}$	unnormalized loss summed over the focus and the non-focus positions
λ	relative weight of non-focus positions
τ	confidence threshold above which a correctly predicted cell is trusted
$d^\star$	length of a shortest path from start to goal in a maze
$\mathcal{U}$, δ_k	non-clue cells of a puzzle; relative residual of the answer state at step k (Appendix C.6.4)
$S_m(a)$	first training step at which method m reaches accuracy a (speedup)
Terminal credit (Appendix C.6.6)	
$J^{(k)}$, $\Phi^{(j,k)}$	$\partial h^{(k)}/\partial\theta$ with $h^{(k-1)}$ fixed; $\partial h^{(j)}/\partial h^{(k)}$
$\mathrm{C}^{(k)}$	cells correct after step k, $\{u : \hat{y}_u^{(k)} = y_u^\star\}$
$\mathcal{R}$	terminal reward $\lvert\mathrm{C}^{(N_{\mathrm{sup}})}\rvert$
$g_{\mathrm{F}}^{(k)}$, $g_{\mathrm{all}}^{(k)}$, $g_{\overline{\mathrm{F}}}^{(k)}$	detached gradients of step k under the focus, dense, and non-focus loss; $g_{\mathrm{all}}^{(k)} = g_{\mathrm{F}}^{(k)} + g_{\overline{\mathrm{F}}}^{(k)}$
$g_{\mathrm{term}}^{(k)}$	credit of step k from the terminal reward (cross-entropy on $\mathrm{C}^{(N_{\mathrm{sup}})}$)
P	Adam preconditioner (Appendix C.6.1)

Table 4: Evaluation metrics.

Task	Metric	Meaning
Sudoku-Extreme	exact match@halt	solved exactly at the step where the halting head fires
Sudoku-Extreme	exact match@max	solved exactly after all N_{sup} steps, ignoring the halting head
Sudoku-Extreme	exact match@halt-or-max	solved exactly at the first step where the halting head fires, or at step N_{sup} if it never fires
Sudoku-Extreme	coverage@halt	the halting head fires within N_{sup} steps
Maze-Hard	valid shortest path	the predicted path is a valid shortest path from start to goal
ARC-AGI-1	pass@2	the task is solved by one of two attempts

A Extended Related Work

A.1 Connection to Latent Recursive Reasoning Models

A recent line of work investigates whether small recurrent architectures, which we refer to as LRRMs (Section 1), can substitute for large language models on algorithmic reasoning tasks such as ARC-AGI, Sudoku, and maze path-finding (Chollet, 2019; Wang et al., 2025). Wang et al. (2025) proposed the Hierarchical Reasoning Model (HRM), an early LRRM that couples two recurrent modules operating at distinct timescales and trains them with a one-step gradient approximation, deep supervision, and adaptive computation time. Jolicoeur-Martineau (2025) argued that neither the hierarchy nor its fixed-point justification is essential, showing that a single two-layer network recursively refining a latent state alongside a candidate answer matches or exceeds HRM at a fraction of the parameter budget. Subsequent extensions either enlarge capacity within the same paradigm (Gao et al., 2025) or strengthen the architectural prior, as in the symbol-equivariant formulation of Freinschlag et al. (2026), which matches prior models with far fewer augmentations and extrapolates zero-shot to Sudoku grid sizes on which both HRM and TRM fail outright. Equilibrium Reasoners (EqR; Huang et al., 2026) intervene on the input side of the recursion: randomized state initialization broadens which attractor basins are reachable during training, and noise injection perturbs the state fed into each iteration to avoid premature convergence to incorrect fixed points, encouraging convergence toward solution-aligned attractors and improving test-time scaling. Deep Improvement Supervision (DIS; Asadulaev et al., 2026) interprets recursive refinement as policy improvement and introduces scheduled intermediate targets, one per fixed corruption level, to guide each step; because the schedule ties each target to a specific step, DIS forgoes adaptive halting and always runs the full schedule. We compare AS-TRM with the two most related of these works, EqR and DIS, in Appendix A.5.

A.2 Connection to Looped Models

Scaling computational depth, either through explicit chain-of-thought reasoning (Wei et al., 2023; Li et al., 2024) or through increased architectural depth (Merrill & Sabharwal, 2025), has been shown to expand the computational capabilities of Transformers, particularly on inherently serial, multi-step reasoning problems. Looped models pursue the second route without adding parameters, by repeatedly applying a shared block. Universal Transformers (Dehghani et al., 2019) first shared weights across depth and paired this with adaptive computation time (Graves, 2017). Huginn (Geiping et al., 2026) and Ouro (Zhu et al., 2026) bring looping to language-model scale, the former to scale test-time computation and the latter to learn how many iterations to spend on each input, while HRM and TRM (Wang et al., 2025; Jolicoeur-Martineau, 2025) apply it with small networks on algorithmic puzzles. Training such models can be characterized along two dimensions: (i) the gradient horizon, which determines how far the gradient of each loss propagates through the iterative computation, and (ii) the loss construction, which determines which parts of the computation contribute to the objective. Along the first, gradients propagate through the fully unrolled computation in Universal Transformers, through a truncated window of the final iterations in Huginn, and only within each thinking step, with the recurrent state detached across thinking-step boundaries, in HRM and TRM. Along the second, Universal Transformers and Huginn supervise only the final state, whereas Ouro, HRM, and TRM combine losses at multiple depths, with every output unit weighted independently of the model's per-unit progress. Our method operates on the second dimension under a step-wise horizon, adapting which units enter the objective at each depth according to the model's current predictions.

A.3 Connection to Curriculum Learning

Curriculum learning orders or reweights supervision according to what the learner can currently absorb. Bengio et al. (2009) introduced the idea in its modern form, framing it as a continuation method that improves both convergence speed and the quality of the minima reached under non-convex objectives. Since an external difficulty measure is rarely available, self-paced learning (SPL) (Kumar et al., 2010) instead derives the ordering from the learner's own current state. Most subsequent work operates at the granularity of whole training examples, deciding which samples to keep or defer: prioritizing points that are learnable, worth learning, and not yet learnt (Mindermann et al., 2022), or using per-example confidence and its variability across epochs to separate easy, ambiguous, and

hard regions of a dataset (Swayamdipta et al., 2020). A smaller body of work moves below the example level: Lin et al. (2025) exclude individual easy tokens that contribute little gradient signal, and Amani et al. (2026) vary how much of the target sequence is revealed for each sample according to past reward. In all of these cases, however, the target itself remains fixed. Closest to our setting, Goldberg & Nivre (2013) observe that training against a single canonical derivation is unnecessarily restrictive when many derivations are valid, and derive a dynamic oracle that supplies correct targets from whatever state the model actually reaches, though without any notion of progression over training. We combine the two: the target is determined dynamically from the model's current predictions, and the supervised region expands over training as the model becomes able to absorb it.

A.4 Connection to Active Learning

Active learning decides, at each round, which examples have their labels revealed to the model (Settles, 2009). Our method can be viewed as an online, unit-level form of active learning. All labels are available, but at each thinking step only the labels of units in the focus set enter the loss, and the remaining labels do not participate in that step's update. The two settings differ in the scarce resource. Active learning is constrained by annotation cost, whereas reasoning models are constrained by their stage-wise capacity relative to the complexity of the complete solution of the current task: when the recurrent state is detached across steps, a target that requires more computation than one step provides cannot be used effectively. The selection criteria also differ. Classical uncertainty sampling favors the points on which the model is least certain (Lewis & Gale, 1994), whereas we exclude currently unreachable units even when they are the most uncertain. This choice is consistent with two findings from the active learning literature. Uncertainty-based selection can repeatedly pick outliers that the model is unable to learn, which cancels its benefit (Karamcheti et al., 2021). When the model's competence is low relative to the task, selecting typical, easy points outperforms selecting uncertain ones (Hacohen et al., 2022). Our focus set is a dynamic version of this principle: reachability is determined by the model's current predictions, and the set of revealed labels expands to harder units as the model improves.

A.5 Connection to Closely Related Work

EqR and DIS are the two closest works, and each shares one part of our goal with AS-TRM. We compare them along what they change, what they aim for, and where we differ.

Equilibrium Reasoners (EqR). EqR (Huang et al., 2026) views a recursive model as a learned dynamical system and asks when extra test-time iterations help. Its answer is that the model must learn task-conditioned attractors: stable fixed points of the latent state that coincide with valid solutions. To shape these attractors, EqR perturbs the *input side* of the recursion during training, with randomized initial states (so that solution-aligned basins are reachable from many starting points) and noise injected into every iteration (so that trajectories do not settle in spurious fixed points early). At test time, it scales along depth and breadth, and uses the fixed-point residual to select among restarts.

AS-TRM does not touch the input side. The initial state and the recursion are those of TRM, no noise is injected, and the only change is which output positions each thinking step is supervised on. Our method is therefore different from EqR in mechanism, but we observe behavior consistent with EqR: the recursion keeps decreasing and converges to a small number and additional test-time iterations continue to improve accuracy. Following EqR, we measure the relative residual of the answer state when the recursion runs far beyond the training budget (Appendix C.6.4). Under AS-TRM the residual keeps falling and exact match keeps improving (Figure 3(c, d)), whereas TRM keeps a high residual and its exact match decays. The recursion under AS-TRM thus settles for most puzzles and additional iterations become useful, which is the behavior EqR sets out to induce. We also keep adaptive halting, as EqR does, and Appendix C.6.5 shows that the halting step tracks puzzle difficulty.

The two methods act on different parts of training. EqR changes the input side of the recursion; AS-TRM changes only which positions are supervised.

Deep Improvement Supervision (DIS). DIS (Asadulaev et al., 2026) interprets each recursion step of TRM as a policy-improvement step, and trains it against a sequence of intermediate targets $y^\dagger_1, \ldots, y^\dagger_{N_{\sup}} = y^\star$ produced by a corruption schedule with decreasing noise. It uses a small fixed number of steps ($N_{\sup} = 6$), conditions the model on the step index, and drops the halting head. It was proposed and evaluated mainly on ARC-AGI and N-Queens; on Sudoku, its authors report only an MLP backbone, where it matches TRM but learns more slowly and suggests that the generic corruption targets may not fit that task.

AS-TRM shares DIS's diagnosis: the final target is a poor signal at every depth, and step-specific supervision gives better credit assignment. The two differ in how the step-specific target is chosen:

- *Open loop versus closed loop.* DIS fixes the target of step k in advance as a function of k alone, so every puzzle follows the same schedule. AS-TRM chooses the supervised positions from the model's current prediction (Table 7), so the supervision follows how far each puzzle has actually progressed. A hard puzzle stays at a small focus set for more steps.
- *Where the label is used.* DIS trains the model toward corrupted solutions at early steps, so the target at a position that is not yet revealed is a wrong or input value. AS-TRM always uses the true label, and decides only *whether* to supervise a position. Positions outside the focus set are down-weighted rather than pushed toward a wrong value. The random-M ablation (Table 12), which supervises an arbitrary subset of positions of the same size, recovers only part of the gain (72.7 versus 77.6).
- *Halting.* A schedule that ends at step $N_{\sup}$ leaves no target before the last step, so DIS cannot use a halting head (Appendix C.3.2). The focus set has no such end point, so AS-TRM keeps TRM's halting unchanged, and the number of steps adapts to difficulty.

Empirically, AS-TRM outperforms our DIS reproduction on Sudoku-Extreme and Maze-Hard at every recursion depth we tried, and is on par with the reported DIS number on ARC-AGI-1 (Table 2). This is consistent with the DIS authors' remark that a generic corruption schedule need not match the structure of Sudoku, and with the schedule-free choice of positions being what helps there. DIS casts recursive supervision as policy improvement, through an advantage margin between the pre- and post-reasoning policy, whereas our analysis (Appendix C.6.6) asks how well the detached gradient on the cells the model can currently get right agrees with the credit of the terminal outcome. We regard them as different ways of attacking the same credit-assignment problem, not as competing explanations.

B LATENT RECURSIVE REASONING MODELS (LRRM)

B.1 ARCHITECTURE COMPARISON BETWEEN HRM AND TRM

Table 5 compares the loop structure of HRM and TRM. HRM uses two modules with separate parameters: the low-level module f_L updates z_L L times for every update of z_H by the high-level module f_H, the answer is decoded from z_H, and the halting head is trained with Q-learning. TRM replaces both modules with a single two-layer network g_θ: it updates the latent state z from (x, y, z) L times and then the answer y from (y, z) once, the answer is decoded from y, and the halting head is trained with binary cross-entropy. In both models one thinking step, written $G_\theta^{H,L}$ in Section 3, repeats this cycle H times, and the recurrent states are detached between thinking steps (Equation (1)). Within a step, the first $H-1$ cycles run without gradients; TRM backpropagates through the full last cycle, whereas HRM uses a one-step gradient through only the last low-level and high-level updates. Counting the layers applied in one thinking step gives the effective depth $D = H(L+1)\, n_{\text{layers}}$ reported in Table 2, with $n_{\text{layers}} = 4$ for HRM and 2 for TRM.

Table 5: Loop structure of HRM and TRM.

Loop		Count	HRM	TRM
Inner	low level	L	$z_L \leftarrow f_L(z_L, z_H, x)$	$z \leftarrow g_\theta(x, y, z)$
	high level	H	$z_H \leftarrow f_H(z_H, z_L)$	$y \leftarrow g_\theta(y, z)$
Outer	deep supervision	N_{sup}	z_L, z_H detached and reused	y, z detached and reused
Parameters			θ_L and θ_H	θ
Answer decoded			by f_O from z_H	from y
Halting term			$f_Q(z_H)$, Q-learning	$f_Q(y)$, BCE
Forward, one thinking step			$\big(f_H \circ f_L^{\circ L}\big)^{\circ H}$	$\big(g_\theta \circ g_\theta^{\circ L}\big)^{\circ H}$
Backward path			$f_H \circ f_L$	$g_\theta \circ g_\theta^{\circ L}$
Layers per module application			4	2
Forward depth per thinking step			$4H(L+1)$	$2H(L+1)$

B.2 UNROLLED EXPERIMENT: REMOVING THE STOP-GRADIENT

TRM detaches the latent state between thinking steps, so with detach, step k receives only the immediate credit of its own loss (Proposition 1). We test removing this stop-gradient. All N_{sup} thinking steps run in one forward pass with the state carried across steps without detaching, so a loss at step j reaches every earlier step $k \le j$ through $\Phi^{(j,k)}$. Let $\ell^{(k)}$ be the cross-entropy of the answer at step k and K_i the halting step of sample i. We backpropagate two choices of losses, which differ in the credit that step k receives (notation of Appendix C.6.6).

- *Terminal credit.* Only the answer at the halting step is supervised, so step $k \le K_i$ receives the credit of the terminal loss (cf. Equation (21)),

$$g^{(k)} = J^{(k)\top} \Phi^{(K_i,k)\top} \nabla \ell^{(K_i)}\big(h^{(K_i)}\big). \tag{8}$$

- *Credit-to-go.* Every step up to K_i is supervised, so step k receives its immediate credit plus the delayed (future) credit of all later steps (Equation (20)),

$$g^{(k)} = J^{(k)\top} \sum_{j=k}^{K_i} \Phi^{(j,k)\top} \nabla \ell^{(j)}\big(h^{(j)}\big). \tag{9}$$

The halting head is trained with $\mathrm{BCE}\big(q_i^{(k)}, \mathbf{1}[\hat{y}_i^{(k)} = y_i^\star]\big)$ at every step up to K_i. With credit-to-go, this loss is backpropagated through the trajectory like the cross-entropy. With terminal credit, the halting head reads a detached state, so it adds no credit to earlier steps. Other settings follow the Sudoku-Extreme recipe.

Table 6 shows that unrolling hurts with either credit. At $N_{\text{sup}} = 16$, exact match@halt-or-max with credit-to-go plateaus at 44.8%, about 15 points below TRM. With terminal credit it peaks at 42.9% at 26k steps and then declines slowly, about 17 points below TRM. It learns faster early on, but stalls

after the first quarter of training. At $N_{\text{sup}} = 8$, credit-to-go collapses to memorization after 20k steps.

Table 6: Removing the stop-gradient between thinking steps on Sudoku-Extreme, $(H, L) = (1, 2)$. The terminal-credit variant is a single run, and we report its peak over evaluations.

Training	N_{sup}	Exact match@halt-or-max (%)
TRM, with detach	16	59.9
Unrolled, credit-to-go	16	44.8
Unrolled, terminal credit	16	42.9
Unrolled, credit-to-go	8	30.7

C Experimental Details

C.1 Task Examples

C.1.1 Sudoku-Extreme

Each puzzle is a 9×9 grid with some given digits (clues), and every empty cell must be filled so that each row, column, and 3×3 box contains each digit exactly once. Figure 6 solves one test puzzle step by step: at each step, every cell whose value follows directly from the digits already on the board is filled, and the cells filled in one step make new cells follow in the next. This is the kind of progress that the frontier of AS-TRM tracks along thinking steps. Most Sudoku-Extreme puzzles are harder than this example: such direct filling alone solves only 2,271 of the first 20,000 test puzzles.

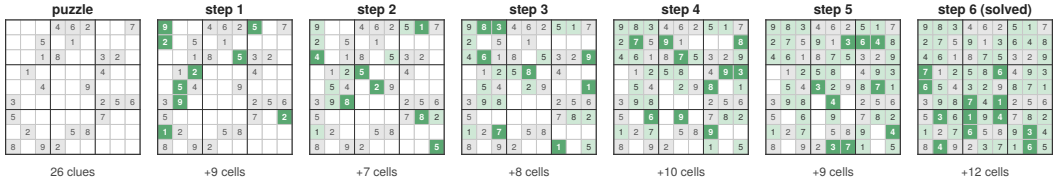


Figure 6: Solving a Sudoku-Extreme test puzzle step by step. Gray cells are clues, dark green cells are filled at the current step, and light green cells were filled at earlier steps.

C.1.2 Maze-Hard

Each input is a 30×30 grid with walls, a start cell S, and a goal cell G, and the target is a shortest path from S to G. A maze usually has many shortest paths: wherever a path can pass an obstacle on either side, or make the same moves in a different order, with the same number of steps, both choices are shortest, and these choices multiply along the path. Every one of the 1,000 test mazes has more than one shortest path, and the median maze has about 3.8 million. The dataset, however, labels only one of them; the shortest path that differs most from the label differs in a median of 34 cells. Figure 7 shows ten of the roughly 3×10^{11} shortest paths of one test maze; all have length 110, and only the first is labeled. For this reason we evaluate valid shortest paths rather than exact match against the label (Section 5), and AS-TRM supervises the shortest path most consistent with the current prediction (Appendix C.2.2).

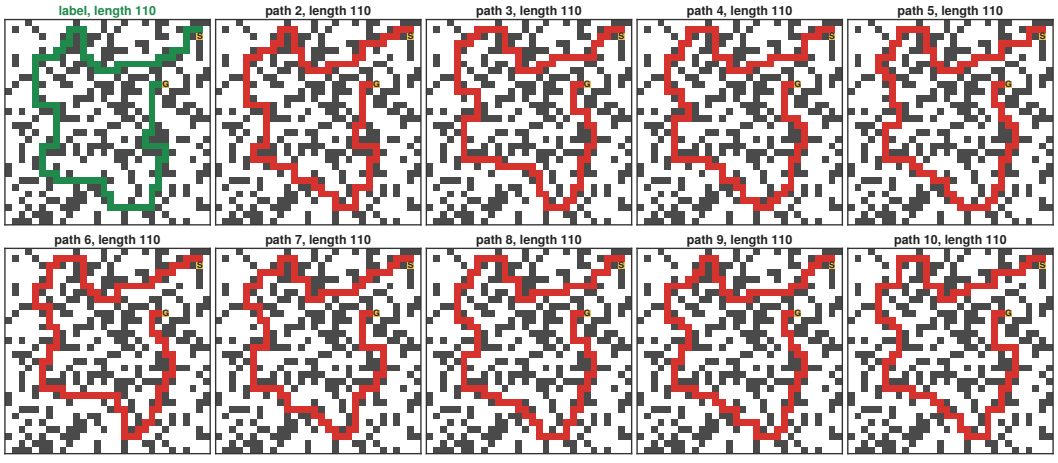


Figure 7: Ten different shortest paths from S to G in one Maze-Hard test maze, all of length 110. Dark cells are walls. The dataset labels only the first (green); the other nine are equally valid solutions.

C.1.3 ARC-AGI-1

Each task gives a few demonstration input–output pairs, and the model must infer the underlying transformation and apply it to a test input. In the task of Figure 8, the 3×3 input is expanded into a 9×9 output by placing a copy of the input wherever the input has a colored cell. Every output cell depends on the whole input through the transformation, so progress is naturally judged per puzzle rather than per cell, which is why AS-TRM selects at the puzzle level on ARC-AGI-1 (Appendix C.2.3). We use 300 augmentations per task.

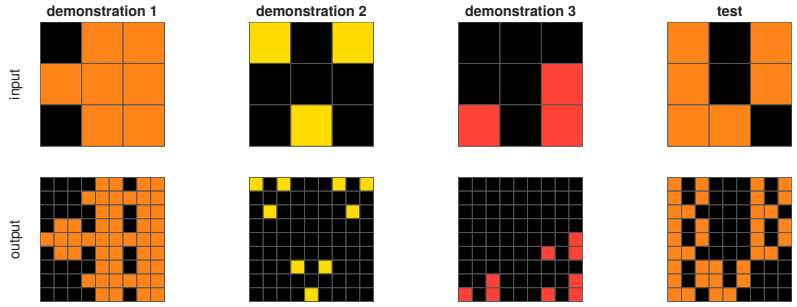


Figure 8: An ARC-AGI-1 training task (007bbfb7): three demonstration pairs and the test pair.

A different kind of task is shown in Figure 9: the input contains a single object, and the output is the same object moved down by one cell and recolored from light blue to red. Here the rule combines a spatial shift with a color change, whereas the task in Figure 8 builds the output by tiling.

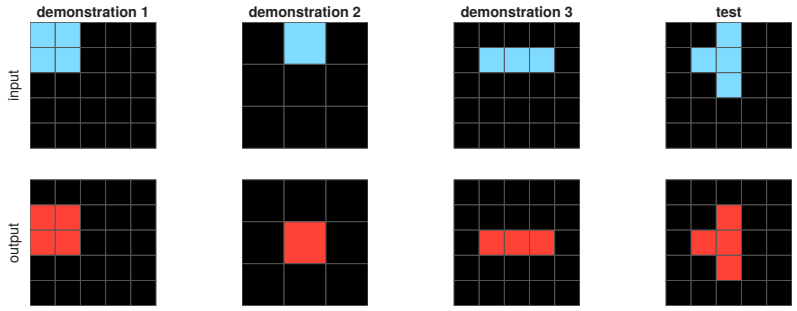


Figure 9: An ARC-AGI-1 training task (a79310a0): three demonstration pairs and the test pair.

C.2 AS-TRM: FOCUS-SET CONSTRUCTION

Table 7 gives the selection rules and hyperparameters for each task; the subsubsections below describe each construction in detail.

Table 7: Task-specific selection rules and hyperparameters. τ is the confidence threshold; the frontier budget controls $|\mathrm{R}_t^{(k)}|$; N_{unsolved} denotes the number of unsolved puzzles in the batch.

Task	Trusted-set rule	Frontier construction	Frontier budget	τ	λ
Sudoku-Extreme	Correctly predicted cells with confidence $\geq \tau$.	Apply naked-single and hidden-single constraint propagation from the trusted cells; select newly derivable cells.	$\lvert\mathrm{R}_t^{(k)}\rvert \leq 20$	0.95	0
Maze-Hard	Cells on a shortest path that are predicted as path with confidence $\geq \tau$.	Extend the trusted chains from the start and from the goal by w distance layers along the shortest-path cells.	$w = \mathrm{round}(0.2d^\star)$ layers per end	0.9	0
ARC-AGI-1	Puzzles with every output cell predicted correctly.	Rank unsolved puzzles by mean per-cell cross-entropy; select the lowest-loss puzzles.	$\lvert\mathrm{R}_t^{(k)}\rvert \approx 0.3N_{\mathrm{unsolved}}$	—	0.11

C.2.1 SUDOKU-EXTREME

The trusted set $\mathrm{T}_t^{(k)}$ contains the cells that are predicted correctly with confidence at least τ. Starting from the clues and the trusted cells, we apply naked-single and hidden-single propagation. The frontier $\mathrm{R}_t^{(k)}$ contains the newly derived cells, capped at $|\mathrm{R}_t^{(k)}| \leq 20$ per puzzle.

C.2.2 MAZE-HARD

Maze-Hard (Wang et al., 2025) records one shortest path per maze, although a maze can admit exponentially many valid shortest paths from start to goal. Let $d_S(u)$ and $d_G(u)$ be the shortest-path distances of a cell u from the start and the goal, $d^\star$ the length of a shortest path, and $\mathrm{SP} = \{u : d_S(u) + d_G(u) = d^\star\}$ the cells that lie on at least one shortest path. We therefore select, at every step, the shortest path $\pi^\star$ with maximum overlap with the high-confidence predictions. It is found by dynamic programming over the cells of SP, ordered by d_S, and maximizes the number of cells that are predicted as path with confidence at least τ; ties are broken by a fixed neighbor order. Because $\pi^\star$ is chosen from the model's predictions rather than fixed by the dataset label, the target stays consistent with the input under flip and rotation augmentation.

The trusted set $\mathrm{T}_t^{(k)}$ contains the cells of SP that are predicted as path with confidence at least τ. The frontier is defined by distance layers from both ends of the path. Let e_S (e_G) be the largest d_S (d_G) that is reached from the start (goal) through a 4-connected chain of trusted cells, and let $w = \mathrm{round}(0.2\,d^\star)$. The focus set contains the cells of SP in the window $\{u \in \mathrm{SP} : d_S(u) \leq e_S + w \text{ or } d_G(u) \leq e_G + w\}$. It contains the two trusted chains, and the frontier $\mathrm{R}_t^{(k)}$ is the rest of the window, i.e., it extends the chains by w layers, at most $0.2\,d^\star$ layers per end. A trusted cell that is not connected to an end and lies outside the window is not supervised.

The supervised set is larger than the focus set. Cells that lie on no shortest path carry no ambiguity about which shortest path is chosen, so all wall cells and all open cells outside SP are always supervised as fixed targets. The only cells that are deferred are the cells of SP outside the window. The target of a supervised cell is the path token if it lies on $\pi^\star$ and its input token otherwise, with start and goal kept; in particular, cells of SP inside the window that are not on $\pi^\star$ are supervised as open cells. The per-sample loss is the mean over the supervised cells, i.e., Equation (5) with $\lambda = 0$ for this supervised set.

C.2.3 ARC-AGI-1

ARC-AGI-1 predictions are governed by a global transformation, so we select at the puzzle level. The trusted set $\mathrm{T}_t^{(k)}$ contains the puzzles whose output is predicted entirely correctly at step k. The

frontier $\mathrm{R}_t^{(k)}$ contains roughly $0.3N_{\text{unsolved}}$ of the unsolved puzzles with the lowest mean per-cell cross-entropy. Non-focus puzzles are weighted by $\lambda = 0.11$.

C.3 BASELINES

We compare against TRM, self-paced learning (SPL) (Kumar et al., 2010), and Deep Improvement Supervision (DIS) (Asadulaev et al., 2026). AS-TRM selects what to supervise within a sample, along recursive depth. SPL selects whole samples, along training time. DIS keeps the full sample but replaces the target at each thinking step by a scheduled intermediate target.

C.3.1 SELF-PACED LEARNING (SPL)

At optimizer step t of a $t_{\max}$-step run, SPL ranks the samples in the batch by their error rate, i.e., the fraction of wrong cells, and keeps the easiest fraction $p(t)$. The selection is recomputed at every supervision step k from the error rate of the step-k prediction, so a sample that is dropped at one thinking step can be kept at another; a dropped sample receives no answer-loss gradient at that step. The kept fraction grows linearly:

$$p(t) = p_{\text{start}} + (1 - p_{\text{start}}) \min\left(1, \frac{t}{\rho\, t_{\max}}\right). \tag{10}$$

We use $p_{\text{start}} = 0.3$ and $\rho = 0.5$ on Sudoku-Extreme and Maze-Hard without further tuning on these datasets, and $p_{\text{start}} = 0.6$ and $\rho \approx 0.93$ on ARC-AGI-1. On ARC-AGI-1, we also tried $p_{\text{start}} = 0.3$, but it barely learns when training from scratch, so we use 0.6. On Sudoku-Extreme, $p_{\text{start}} = 0.3$ is close to the fraction of given cells in a puzzle (≈ 0.31), i.e., the cells that AS-TRM trusts at the start of training.

C.3.2 DEEP IMPROVEMENT SUPERVISION (DIS)

DIS supervises thinking step k against an intermediate target $y_k^\dagger$ instead of the full solution $y^\star$, where $y_{N_{\text{sup}}}^\dagger = y^\star$ and the targets move monotonically from the puzzle input toward the solution. We reimplement it on Sudoku-Extreme and Maze-Hard with the same TRM backbone, data, optimizer, and training budget as TRM, and change only the supervision. The intermediate targets follow a linear path from the input to the solution: for a puzzle with m positions that differ between input and solution, $y_k^\dagger$ reveals the solution values at a random subset of $\lfloor k\, m / N_{\text{sup}} \rfloor$ of these positions and keeps the input values elsewhere. The step index is provided to the model through a learned embedding. DIS has no halting head, so each sample always runs the full N_{sup} thinking steps and @halt-or-max coincides with @max. This is inherent to its design: the intermediate targets are corrupted solutions on a schedule that ends at the last step, so halting has no target before the last step. We follow the setting of the DIS paper for the schedule. On ARC-AGI-1 we quote the number reported by the DIS authors. Table 8 lists the full configuration of our DIS runs. Everything not listed there is shared with TRM (Tables 9 to 11): hidden size 512, 8 attention heads, 2 layers, MLP expansion 4, RoPE, stablemax cross-entropy, Adam-Atan2 with $(\beta_1, \beta_2) = (0.9, 0.95)$, learning rate 10^{-4} with 2k warmup steps, EMA rate 0.999, weight decay 1.0, and puzzle embeddings of length 16 and dimension 512 with the same learning rate and weight decay. Each configuration is trained with 3 seeds, and we evaluate every 5000 epochs on Sudoku-Extreme and every 100 epochs on Maze-Hard.

Table 8: Configuration of the DIS runs. Rows in the lower block differ from TRM.

	Sudoku-Extreme	Maze-Hard
Recursion depth (H, L)	$(1,2)$, $(2,4)$, $(3,6)$	$(1,2)$
N_{sup}	6 (as in the DIS paper)	
Global batch size	768	128
Training epochs	50k	10k
LR schedule	constant	cosine to $0.5\times$
Intermediate target	linear path from input to solution	

C.4 HYPERPARAMETERS AND COMPUTE

Tables 9 to 11 list the training hyperparameters for Sudoku-Extreme, Maze-Hard, and ARC-AGI-1, respectively. All tasks use the TRM architecture with hidden size 512, 8 attention heads, 2 layers, MLP expansion 4, and a maximum of $N_{\text{sup}} = 16$ thinking steps with halting exploration probability 0.1. The recursion depth (H, L) of each run is given in Table 2. TRM, AS-TRM, and SPL use the same hyperparameters, including $N_{\text{sup}} = 16$; AS-TRM only changes which output positions are supervised (Table 7). DIS uses the same backbone, data, and optimizer, but $N_{\text{sup}} = 6$ and the other method-specific settings given in Appendix C.3 (Table 8). ARC-AGI-1 is configured for 60k epochs (≈311k steps), but we compare all methods at step 116,563: training TRM for the full configured 311k steps raises its pass@2 by only about 1 point.

Table 9: Training hyperparameters and compute on Sudoku-Extreme.

Hyperparameter	Value
Global batch size	768
Training epochs	50k
Training steps	65,104
Optimizer	Adam-Atan2 (Everett et al., 2024), $(\beta_1, \beta_2) = (0.9, 0.95)$
Learning rate	10^{-4}, 2k warmup steps
LR schedule	constant
Weight decay	1.0
Puzzle-embedding LR	10^{-4}
Puzzle-embedding weight decay	1.0
EMA rate	0.999
Seeds	5 (3 for DIS)
GPUs	RTX PRO 6000 / RTX 6000 Ada / L40S / A100

Table 10: Training hyperparameters and compute on Maze-Hard.

Hyperparameter	Value
Global batch size	128
Training epochs	10k
Training steps	78,125
Optimizer	Adam-Atan2 (Everett et al., 2024), $(\beta_1, \beta_2) = (0.9, 0.95)$
Learning rate	10^{-4}, 2k warmup steps
LR schedule	cosine to $0.5\times$
Weight decay	1.0
Puzzle-embedding LR	10^{-4}
Puzzle-embedding weight decay	1.0
EMA rate	0.999
Seeds	5 (3 for DIS)
GPUs	RTX PRO 6000 / RTX 6000 Ada / L40S / A100

Table 11: Training hyperparameters and compute on ARC-AGI-1. It is configured for 60k epochs (≈311k steps); we compare all methods at step 116,563.

Hyperparameter	Value
Global batch size	768
Training epochs	≈22.5k
Training steps	116,563
Optimizer	Adam-Atan2 (Everett et al., 2024), $(\beta_1, \beta_2) = (0.9, 0.95)$
Learning rate	10^{-4}, 2k warmup steps
LR schedule	constant
Weight decay	0.1
Puzzle-embedding LR	10^{-2}
Puzzle-embedding weight decay	0.1
EMA rate	0.999
Seeds	3
GPUs	4×H100

C.5 Additional Ablations

C.5.1 Sudoku-Extreme Frontier Ablation

Does the frontier matter, or only its size? Table 12 keeps $\mathrm{T}_t^{(k)}$ and the training recipe fixed and changes only which other cells are supervised. *Random M cells* supervises as many cells as AS-TRM at every step, drawn at random from the non-trusted cells; *inverse frontier* supervises exactly the cells AS-TRM discards. Matching the number of cells recovers only part of the gain, and inverting the frontier removes it entirely: the frontier is the mechanism, not a budget heuristic.

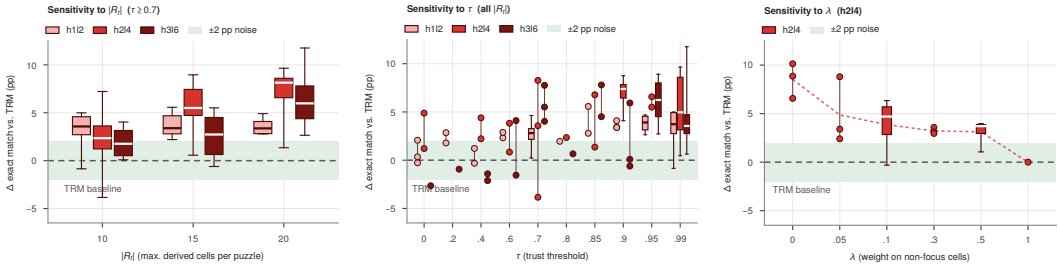
Table 12: Frontier ablation on Sudoku-Extreme at $(H, L) = (2, 4)$. All variants share the trusted set $\mathrm{T}_t^{(k)}$ and $N_{\mathrm{sup}} = 16$, and differ only in the frontier $\mathrm{R}_t^{(k)}$.

Frontier $\mathrm{R}_t^{(k)}$	Exact match@halt-or-max (%)
TRM (no frontier)	69.7
Random M cells	72.7
Inverse frontier, $\lambda = 0.1$	66.3
Inverse frontier, $\lambda = 0$	0.0
AS-TRM	**77.6**

C.5.2 Sensitivity Analysis on Sudoku-Extreme

AS-TRM has three hyperparameters on Sudoku-Extreme: the trust threshold τ, above which a predicted cell seeds the solver, the frontier budget $|\mathrm{R}_t|$, the maximum number of cells derived per puzzle, and the weight λ on non-focus cells. Figure 10 sweeps $|\mathrm{R}_t|$, τ, and λ at three recursion depths, $(H, L) \in \{(1, 2), (2, 4), (3, 6)\}$.

Performance is best at $|\mathrm{R}_t| \in [15, 20]$ (Figure 10a), where AS-TRM consistently outperforms TRM. Accuracy plateaus for $\tau \geq 0.85$ (Figure 10b), so AS-TRM is robust to its exact value. The gain shrinks as λ grows (Figure 10c) and vanishes at $\lambda = 1$, which recovers the TRM baseline. We use $\tau = 0.95$ and $|\mathrm{R}_t| = 20$ in Table 2.



(a) Frontier budget $|\mathrm{R}_t|$ ($\tau \geq 0.7$). (b) Trust threshold τ (all $|\mathrm{R}_t|$). (c) Non-focus weight λ ($|\mathrm{R}_t| = 20$, $\tau = 0.95$).

Figure 10: Sensitivity of AS-TRM on Sudoku-Extreme. Each panel is shown at three recursion depths (H, L) (h1l2, h2l4, h3l6 in the legend); $\lambda = 1$ is the TRM baseline by definition. Boxes aggregate all runs at each value (median, quartiles, range; fewer than 4 runs shown as points). The vertical axis is the change in exact match over the same-depth TRM baseline; the green band is the ± 2 pp seed noise of the TRM baseline.

C.5.3 Maze-Hard Data Augmentation

The Maze-Hard gains should not be interpreted as an effect of adaptive masking in isolation. Maze admits multiple valid shortest paths, while the dataset provides only one canonical target. Under flip and rotation augmentation, this ambiguity can make the fixed target inconsistent with the transformed input. AS-TRM resolves this by constructing supervision from a model-consistent valid shortest path (Appendix C.2.2). Table 13 shows that neither augmentation alone nor adaptive supervision without augmentation explains the final gain; the improvement arises from their interaction.

Table 13: Maze-Hard ablation. Augmentation alone collapses because of ambiguous A* tie-break targets. Peak over training, 30×30 hard mazes, $(H, L) = (1, 2)$.

Method	Valid shortest path@halt-or-max (%)
TRM	86.8
TRM w/ augmentation	0.1
AS-TRM w/o augmentation	81.5
AS-TRM	93.2

C.5.4 Results under @max Evaluation

Table 2 reports @halt-or-max. Table 14 repeats it under @max, which always runs all N_{sup} thinking steps and ignores the halting head, the protocol used by prior work. As in the main table, each entry is the peak over training, averaged over seeds, and it comes from the same experiments as Table 2; the footnotes list the exceptions. DIS has no halting head, so its @max equals its @halt-or-max entry. For ARC-AGI-1, we report pass@2 at step 116,563, as in the main table. Figure 11 shows the corresponding training curves.

Table 14: Main results under @max evaluation. Best result per column in bold, second best underlined. Small numbers in parentheses are standard deviations across seeds. Δ is the difference between AS-TRM and the best other method in each column.

	Sudoku-Extreme (exact match)			Maze-Hard (valid shortest path)	ARC-AGI-1 (pass@2)
(H, L)	$(1, 2)$	$(2, 4)$	$(3, 6)$	$(1, 2)$	$(3, 4)$
TRM	<u>62.9</u> (3.2)	<u>67.8</u> (4.4)	<u>72.2</u> (2.1)	89.0 (2.5)	40.8 (1.0)
TRM + SPL	57.6 (6.3)	63.1 (5.3)	69.7 (4.5)	<u>89.1</u> (1.0)	38.1 (0.2)
DIS	38.0 (1.1)	64.8 (2.5)	69.2 (1.6)	72.0 (2.2)	<u>41.3</u>*
AS-TRM	**66.1** (7.2)	**73.5** (11.8)	**78.5** (2.1)	**95.6** (0.5)	**41.4** (0.5)
Δ	+3.2	+5.7	+6.3	+6.5	+0.1

All other Sudoku-Extreme and Maze-Hard entries use five seeds, except DIS and Maze-Hard SPL, which use three.
* Taken from Asadulaev et al. (2026).

Under @max, AS-TRM is best in all five settings. Compared with @halt-or-max, the @max entries of TRM and AS-TRM vary more across seeds on Sudoku-Extreme $(1, 2)$ and $(2, 4)$, since without the halting head a run can overwrite a correct answer in later steps.

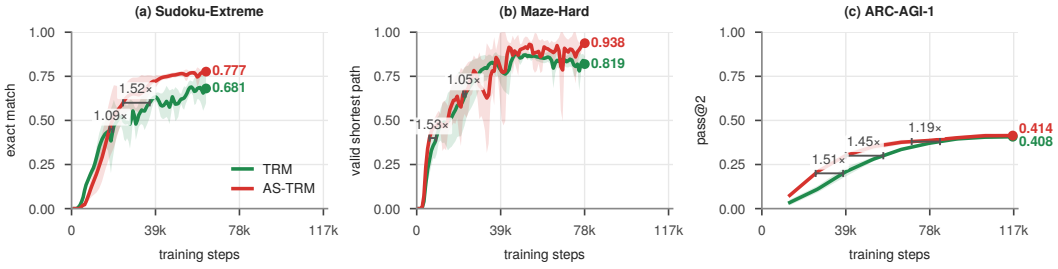


Figure 11: Training curves under @max, the counterpart of Figure 2: mean and standard deviation over seeds, with Sudoku-Extreme at $(H, L) = (3, 6)$. Labels at the end of each curve are final values. Brackets mark the training steps at which each method first reaches a threshold, shown only where AS-TRM is faster.

C.6 Details for Experimental Analysis

C.6.1 Gradient Interference

Let g_{F} and $g_{\overline{\mathrm{F}}}$ denote the gradients of the focus-set and non-focus-set losses. To quantify how non-focus supervision interferes with learning, we measure the work done by an update $\Delta\theta$ on the focus loss and on the non-focus loss,

$$W_{\mathrm{F}} = -\langle g_{\mathrm{F}}, \Delta\theta \rangle, \qquad W_{\overline{\mathrm{F}}} = -\langle g_{\overline{\mathrm{F}}}, \Delta\theta \rangle. \tag{11}$$

With the optimizer update

$$\Delta\theta = -\eta P(g_{\mathrm{F}} + g_{\overline{\mathrm{F}}}), \tag{12}$$

where $P \succ 0$ is the Adam preconditioner, the work expands to

$$W_{\mathrm{F}} = \eta\, g_{\mathrm{F}}^{\top} P g_{\mathrm{F}} + \eta\, g_{\mathrm{F}}^{\top} P g_{\overline{\mathrm{F}}}. \tag{13}$$

The first term is the solo progress $W_{\mathrm{F}}^{\mathrm{solo}}$, and the second term is the cross-talk. We define the slowdown factor

$$\kappa = -\frac{g_{\mathrm{F}}^{\top} P g_{\overline{\mathrm{F}}}}{g_{\mathrm{F}}^{\top} P g_{\mathrm{F}}}, \tag{14}$$

which multiplies the number of training steps by $1/(1-\kappa)$. Since $W_{\mathrm{F}} + W_{\overline{\mathrm{F}}} = \eta(g_{\mathrm{F}} + g_{\overline{\mathrm{F}}})^{\top} P(g_{\mathrm{F}} + g_{\overline{\mathrm{F}}}) \geq 0$, the full loss still decreases, so this interference does not show up on standard training curves.

Figure 12 plots, for every pair of thinking steps (i, j), the inner product between the focus-set gradient at step i and the non-focus-set gradient at step j; with $N_{\mathrm{sup}} = 16$ this gives $16 \times 16 = 256$ entries. At $(H, L) = (2, 4)$ after 26k steps, all 256 entries are negative (mean -0.71), indicating that non-focus gradients consistently push shared parameters against the focus gradients. Setting $\lambda = 0$ removes this interference.

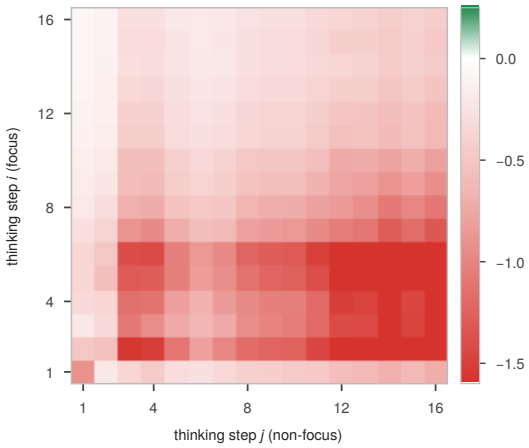


Figure 12: Inner product between focus-set gradients at thinking step i and non-focus-set gradients at step j, Sudoku-Extreme, $(H, L) = (2, 4)$, 26k steps.

C.6.2 Loss Landscape

For puzzle p and thinking step k, let $\mathrm{F}_p^{(k)}$ denote the focus set and $\overline{\mathrm{F}}_p^{(k)}$ its complement. For $S \in \{\mathrm{F}, \overline{\mathrm{F}}\}$, define

$$\bar{\ell}_S(p, k; \theta) = \frac{1}{|S_p^{(k)}|} \sum_{u \in S_p^{(k)}} \mathrm{CE}\big(f_\theta^k(p)_u, y_u^\star\big), \tag{15}$$

where $f_\theta^k(p)_u$ is the step-k prediction for cell u and $y_u^\star$ is the ground truth. Following Li et al. (2018), we visualize

$$\bar{\ell}_S^{(p,k)}(\alpha, \beta) = \bar{\ell}_S(p, k; \theta + \alpha d_1 + \beta d_2) \tag{16}$$

on a 51×51 grid over $[-1, 1]^2$, with filter-normalized random directions d_1, d_2.

As shown in Figure 4, the focus-set surfaces keep a consistent geometry across thinking steps, whereas the non-focus-set surfaces change from step to step. For each puzzle p, we fit a quadratic to the grid at each step $k = 2, \ldots, N_{\text{sup}}$ ($N_{\text{sup}} = 16$; step 1 excluded) and compute the Hessian

$$\text{Hess}_S^{(k)} = \nabla^2_{(\alpha,\beta)} \bar{\ell}_S^{(p,k)}(0,0) \in \mathbb{R}^{2\times 2}, \tag{17}$$

with eigenvalues $\mu_{\min}^{(k)} < \mu_{\max}^{(k)}$. $\mu_{\min}^{(k)} > 0$ indicates a bowl and $\mu_{\min}^{(k)} < 0$ a saddle. The flip rate is

$$\text{fr}^{(p)} = \frac{1}{N_{\text{sup}} - 2} \sum_{k=2}^{N_{\text{sup}}-1} \mathbf{1}\left[\mu_{\min}^{(k)} \mu_{\min}^{(k+1)} < 0\right], \tag{18}$$

the fraction of consecutive steps at which the local geometry switches between bowl and saddle.

We average over the 375 training puzzles (out of a batch of 768) whose non-focus set is non-empty, i.e., at least 3 cells at 4 or more steps. At the 20.8k TRM checkpoint, $\text{fr}_{\text{F}} = 0.03$ and $\text{fr}_{\overline{\text{F}}} = 0.39$: the non-focus surface flips roughly $13\times$ more often. The same pattern holds at 39k (0.05 vs. 0.38) and 65k (0.03 vs. 0.37). The focus set thus provides a depth-consistent optimization signal, whereas the non-focus set alternates between helpful and harmful supervision across steps.

C.6.3 Attention Dynamics on a Single Puzzle

Figure 13 shows, over three matched pairs of checkpoints, that TRM solves puzzles and then unsolves them and that its attention keeps drifting; Figure 14 explains subfigure (a) at the level of a single puzzle. We take one Sudoku-Extreme test puzzle at $(H, L) = (2, 4)$, run both models for 16 thinking steps with halting disabled, and color each cell by how much its attention changes between consecutive steps, $1 - \cos(a_u^{(k)}, a_u^{(k-1)})$, where $a_u^{(k)}$ is the head-averaged attention distribution of cell u at step k. Both models solve this puzzle at step 3, and the attention of both settles by steps 4–8. After step 8, TRM's attention starts changing again across most of the board, and TRM overwrites its solution: 11 cells are wrong at step 12 and 41 at step 16. The attention of AS-TRM stays unchanged, and its answer remains correct through step 16. This is the per-puzzle view of subfigure (b) of Figure 13: under dense supervision the recursion keeps moving after it has reached the answer.

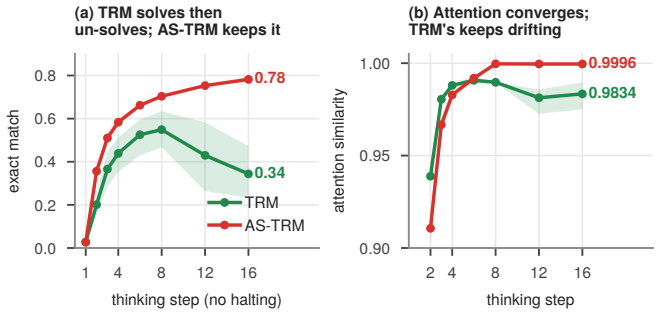


Figure 13: Rollout without halting for 16 thinking steps on 512 Sudoku-Extreme test puzzles, $(H, L) = (2, 4)$, mean over three matched pairs of checkpoints (shaded: range across pairs). (a) Exact match after k steps. (b) Per-cell attention similarity between consecutive steps, median over 28,572 non-clue cells of the same puzzles.

C.6.4 Convergence of the Recursion

Setup of Figure 3(c, d). We use the same pair of $(H, L) = (2, 4)$ checkpoints as Figure 14 and the same 512 Sudoku-Extreme test puzzles. Starting from the learned initial states, each model applies its thinking step 1024 times with halting disabled, 64 times the training budget of $N_{\text{sup}} = 16$. At every step we record the relative residual $\delta_k = \|y_u^{(k)} - y_u^{(k-1)}\| / \|y_u^{(k-1)}\|$ of the answer state of each non-clue cell u, averaged over the non-clue cells of each puzzle, and report the median and interquartile range over puzzles; exact match after step k is computed from the step-k prediction.

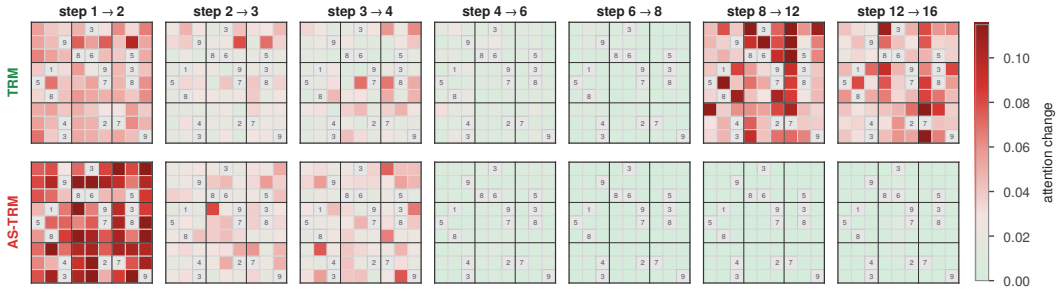


Figure 14: Per-cell attention change between consecutive thinking steps on one Sudoku-Extreme test puzzle, $(H, L) = (2, 4)$, halting disabled. Green cells keep their attention, red cells change it; gray cells are clues. TRM's attention changes again after step 8, when it overwrites the solved board (Figure 13, subfigure (a)); the attention of AS-TRM stays stable.

In the figures, every step is plotted up to step 16. Later steps are sampled at about 40 log-spaced points between step 16 and step 1024, and each plotted value is the average (for the residual, the median) over the steps within 10% of that step index; for example, the point at step 500 averages steps 455 to 550. This only smooths the curves for readability: the end-point labels in the figures are unsmoothed. All curves come from a single pair of checkpoints, so they do not show variability across seeds.

Evidence. Figure 3(c, d) show that the residual of TRM stays high while its state keeps moving after the answer is reached, and its exact match decays. The residual of AS-TRM keeps decreasing, and its exact match keeps improving past the training budget. Under AS-TRM the recursion therefore settles for most puzzles but not for all of them: some puzzles still have a large residual at step 1024, and some are still unsolved. All these results come from a single pair of checkpoints. Figure 15 plots exact match against the median residual: AS-TRM moves toward low residual and high exact match together, whereas TRM stays at high residual.

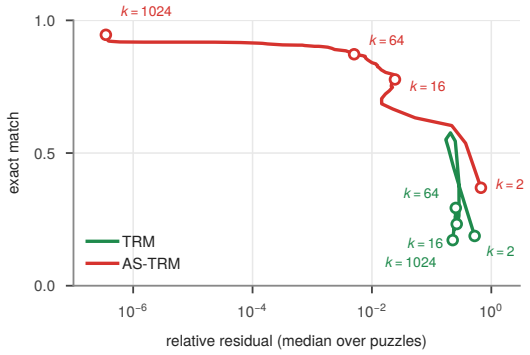


Figure 15: Exact match against the median relative residual, one point per thinking step (same rollout as Figure 3(c, d)). Circles mark steps 2, 16, 64, and 1024.

C.6.5 Halting Adapts to Puzzle Difficulty

Because AS-TRM keeps TRM's halting head unchanged (Section 3), we ask whether the number of thinking steps it uses reflects puzzle difficulty. We label every Sudoku-Extreme test puzzle as easy, medium, or hard with a solving-technique-based difficulty scorer, and split the same 512 test puzzles used in Figures 3, 13 and 14 into 66 easy, 132 medium, and 314 hard puzzles. For the three matched checkpoint pairs of Figure 13, taken at the end of training, we run each model with halting enabled and record, per puzzle, the step at which it halts and whether its answer is then correct. Exact match@halt in this analysis is the fraction of the puzzles of a tier that halt within the budget and are solved at the halting step. It is therefore not the accuracy among halted puzzles, which is close to 0.989 in Figure 3(a). These checkpoints are different from the 3 TRM and 3 AS-TRM

training runs of Figure 3(a, b) (Sudoku-Extreme, $(H, L) = (2, 4)$), and the end-of-training TRM checkpoints are below the peak values of Table 2, so the absolute values here are not comparable to those.

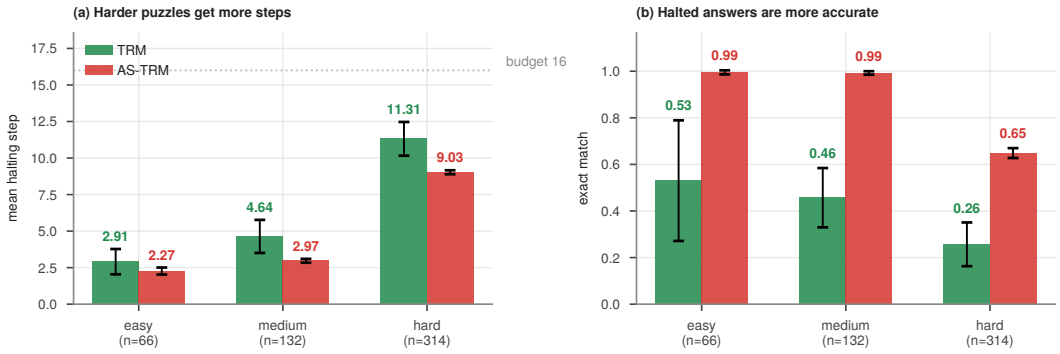


Figure 16: Sudoku-Extreme, by difficulty tier, mean ± sd over the three checkpoint pairs of Figure 13, same 512 test puzzles as Figure 3. (a) Mean halting step (halting enabled). (b) Exact match@halt: fraction of the tier's puzzles that halt and are solved at the halting step.

Figure 16a shows that AS-TRM's halting step increases monotonically with difficulty and is consistent across checkpoints: mean 2.3 steps on easy, 3.0 on medium, and 9.0 on hard puzzles. TRM does not show this pattern as cleanly: mean 2.9, 4.6, and 11.3 steps, with substantially higher variance across checkpoints on every tier (Figure 16a, error bars). Figure 16b shows this is not merely thinking less: AS-TRM's exact match@halt is 0.99 on easy, 0.99 on medium, and 0.65 on hard puzzles, against TRM's 0.53, 0.46, and 0.26, respectively. Preserving the halting head therefore lets AS-TRM allocate computation where it is needed, spending few steps on easy puzzles and more on hard ones, while also answering more accurately at every difficulty level of these checkpoints.

C.6.6 Alignment with the Terminal Credit

Section 5.3.3 asks whether the detached gradient of the focus-cell loss is a better proxy for the unrolled terminal gradient than the dense one. This subsection gives the definitions and the full measurements; it is an analysis of the gradients that detachment leaves, and AS-TRM itself uses no sampling, policy gradient, or reinforcement learning. Notation is listed in Table 3.

Terminal reward and detached credit. A recursive model produces a trajectory of states $h^{(1)}, \ldots, h^{(N_{\sup})}$, and what ultimately matters is the terminal prediction. For cell-decomposable tasks such as Sudoku, let $\mathrm{C}^{(k)} = \{u : \hat{y}_u^{(k)} = y_u^\star\}$ be the cells that are correct after step k, and let the terminal reward be the number of cells correct at the last thinking step,

$$\mathcal{R} = \left|\mathrm{C}^{(N_{\sup})}\right|. \tag{19}$$

Let $h^{(k)} = f_\theta(h^{(k-1)})$ with $f_\theta = G_\theta^{H,L}$ (Equation (1)), $J^{(k)} = \partial f_\theta(h^{(k-1)})/\partial\theta$ at fixed $h^{(k-1)}$, and $\Phi^{(j,k)} = \partial h^{(j)}/\partial h^{(k)}$ for $j \geq k$, with $\Phi^{(k,k)} = I$.

Proposition 1 (immediate and delayed credit)**.**

$$\nabla_\theta \sum_{j=1}^{N_{\sup}} \ell^{(j)}(h^{(j)}) = \sum_{k=1}^{N_{\sup}} J^{(k)\top}\Big[\underbrace{\nabla\ell^{(k)}(h^{(k)})}_{\textit{immediate}} + \underbrace{\textstyle\sum_{j>k} \Phi^{(j,k)\top}\nabla\ell^{(j)}(h^{(j)})}_{\textit{delayed}}\Big], \tag{20}$$

and the detached gradient (Equation (1)*) is* $\sum_k J^{(k)\top}\nabla\ell^{(k)}(h^{(k)})$.

Proof. $\mathrm{d}h^{(j)}/\mathrm{d}\theta = \sum_{k\leq j} \Phi^{(j,k)} J^{(k)}$; substitute into $\nabla_\theta \ell^{(j)} = (\mathrm{d}h^{(j)}/\mathrm{d}\theta)^\top \nabla\ell^{(j)}(h^{(j)})$ and exchange the sums. Detachment sets $\mathrm{d}h^{(j)}/\mathrm{d}\theta = J^{(j)}$. □

Detachment removes the path $h^{(k)} \to h^{(k+1)} \to \cdots \to h^{(N_{\sup})}$, so step k is trained only by its own loss and never observes whether the same update increases $\mathcal{R}$ after further recursion. Which cells to supervise at step k therefore determines how well this local update agrees with the credit that the terminal outcome would have sent.

Gradient alignment. Since $\mathcal{R}$ is not differentiable, we use the cross-entropy on the cells that are correct at the last step, $\ell^{(N_{\sup})}_{\mathrm{C}^{(N_{\sup})}}$ with $\mathrm{C}^{(N_{\sup})}$ held fixed, and backpropagate it to the parameters as used at step k only:

$$g^{(k)}_{\mathrm{term}} = J^{(k)\top} \Phi^{(N_{\sup},k)\top} \nabla \ell^{(N_{\sup})}_{\mathrm{C}^{(N_{\sup})}}, \tag{21}$$

the credit that the terminal reward would have sent to step k without detachment. The detached gradients at step k are

$$g^{(k)}_{\mathrm{F}} = J^{(k)\top} \nabla \ell^{(k)}_{\mathrm{F}^{(k)}}, \qquad g^{(k)}_{\mathrm{all}} = J^{(k)\top} \nabla \ell^{(k)}, \qquad g^{(k)}_{\overline{\mathrm{F}}} = J^{(k)\top} \nabla \ell^{(k)}_{\overline{\mathrm{F}}^{(k)}}, \tag{22}$$

with $g^{(k)}_{\mathrm{all}} = g^{(k)}_{\mathrm{F}} + g^{(k)}_{\overline{\mathrm{F}}}$, all computed on the same TRM checkpoint and differing only in the loss. The intuition of Section 5.3.3 predicts

$$\text{(i)}\ \cos\big(g^{(k)}_{\mathrm{F}}, g^{(k)}_{\mathrm{term}}\big) > \cos\big(g^{(k)}_{\mathrm{all}}, g^{(k)}_{\mathrm{term}}\big), \qquad \text{(ii)}\ \cos\big(g^{(k)}_{\overline{\mathrm{F}}}, g^{(k)}_{\mathrm{term}}\big) < 0. \tag{23}$$

We use $N_{\sup} = 16$, $k \in \{2, 4, 8, 12\}$, and 3×768 training puzzles, for $(H, L) = (1, 2)$ at 5% and 10% of training with three seeds, and $(H, L) = (2, 4)$ at 2, 4, 6, 8, and 10% with one seed. Prediction (i) holds in $22/24$ settings for $(1, 2)$ and in $18/20$ for $(2, 4)$; all exceptions are ties, at $k \leq 4$ for $(1, 2)$ and at 2% of training for $(2, 4)$ (Figure 5). Prediction (ii) holds for $(1, 2)$ in all but two near-zero settings ($\cos(g^{(k)}_{\overline{\mathrm{F}}}, g^{(k)}_{\mathrm{term}}) \in [-0.55, 0.03]$) and for $(2, 4)$ from 6% of training on ($[-0.86, -0.01]$), increasingly so at larger k. For $(2, 4)$ at 2–4% it fails, with values between -0.06 and 0.12: before the model has learned the rules, supervising any cell still helps later steps. Across training (Figure 17), the separation appears from about 6% of training on, peaks around 8–10%, and vanishes once $g^{(k)}_{\mathrm{term}}$ exceeds the detached gradient by a large factor, where all cosines approach zero. Restricted to the cells outside the focus set at step k but correct at step $N_{\sup}$, which still require later reasoning and make up about 4% of the cells, $\cos(g^{(k)}_{\overline{\mathrm{F}}}, g^{(k)}_{\mathrm{term}}) \in [-0.41, -0.03]$ and $\cos(g^{(k)}_{\mathrm{F}}, g^{(k)}_{\mathrm{term}}) \in [0.15, 0.50]$ on another 3×768 puzzles at 5–6% of training. These observations show that, early in training, the focus-cell loss gives a local update direction better aligned with terminal credit, and the non-focus part points against it.

Limits. The check is first-order and local. Without restricting the reference to correct cells, the two detached gradients are about equally aligned, since on cells that stay wrong the cross-entropy also rewards spreading probability, which does not change $\mathcal{R}$; restricting instead to puzzles that are fully correct at the end leaves almost no non-focus cells, so the two losses nearly coincide. The check is informative only early in training, because later $g^{(k)}_{\mathrm{term}}$ exceeds the detached gradient by one to three orders of magnitude. The cell-wise reward is exact for decomposable tasks such as Sudoku; on ARC-AGI-1 the natural unit is coarser and derivability is approximated at the puzzle level. None of these results by itself establishes higher final accuracy or faster training, which are evaluated in the main experiments.

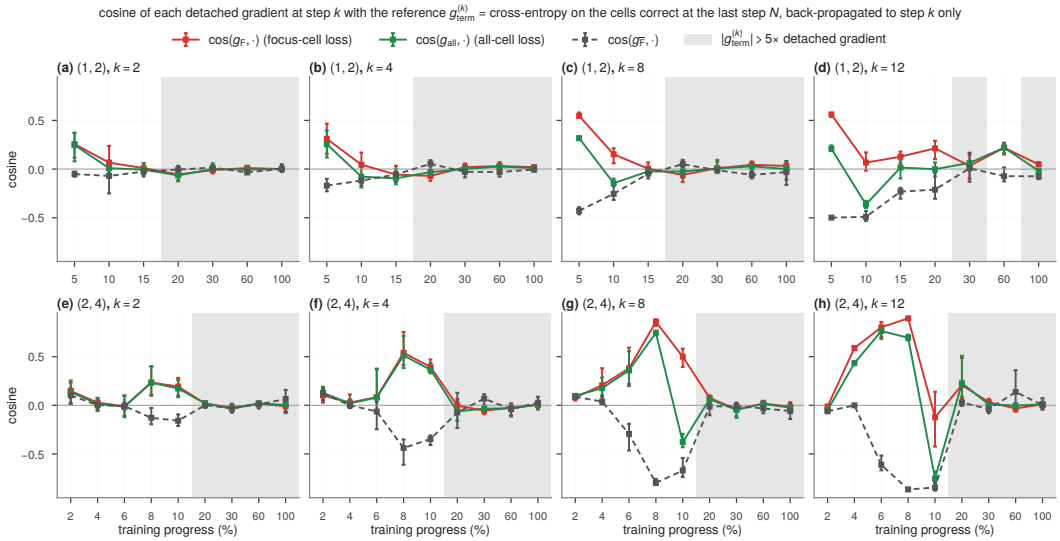


Figure 17: The comparison of Figure 5 across training, for $(H, L) = (1, 2)$ (seed 1, 5–100% of training) and $(2, 4)$ (2–100%), at thinking steps $k \in \{2, 4, 8, 12\}$. Grey background: $g^{(k)}_{\text{term}}$ exceeds the detached gradient by more than a factor of 5, where the comparison is not informative. Error bars show the range over three batches of 768 puzzles.

D FAILED EXPERIMENTS

On ARC-AGI-1, TRM prepends a learned puzzle embedding to every input, with a separate embedding for each (task, augmentation) pair. This large table lets the model heavily overfit the training set: training exact match is near 100%, yet evaluation pass@2 remains under 40%, even with heavy augmentation and majority voting. Once nearly every training puzzle is solved, AS-TRM method has little left to select, since there is no room to grow. To curb this overfitting, we let augmentations share an embedding, across rotations or across color permutations, but this hurt the baseline itself, lowering its accuracy or slowing its learning. A natural direction for future work is to remove the puzzle embedding altogether, for example by conditioning the model on the demonstration pairs in context, so that the transformation is inferred rather than stored.