

Tiny in Parameters, Not in Compute: What a Recursive Reasoning Model's Puzzle Score Credits to Recursion, Why the Ablations Split by Task, and the Matched Comparisons That Would Assign the Rest

Pranay Mahendrakar

SONYTECH · ORCID 0009-0003-7224-029X

pranaymahendrakar@sonytech.in · mahendrakarpranay@gmail.com · +91 6361723454

Abstract. Recursive reasoning models, which apply one small network many times to a latent state, report accuracies on Sudoku, maze and ARC-AGI puzzles above those of most frontier language models in their comparison tables, with 5 to 27 million parameters. The result is usually credited to recursion: iterating a latent state is said to buy the depth that reasoning needs. A 2025-2026 line of ablations, mechanistic studies and controlled re-evaluations disputes that credit. In this analysis the author reviews and synthesises published results to show that the disagreement dissolves once two things are separated: what the word recursion names, and which task the score comes from. Recursion names three functions - computation through iterated depth, regularisation through parameter sharing, and a source of candidate trajectories for a learned selector - and a reported score also carries per-task identity conditioning, training on the evaluation tasks' demonstrations, augmentation with voting, and an optimisation recipe. On Sudoku-Extreme the evidence credits iteration directly: one-pass models of equal size solve almost no puzzles, and running the recursion past its training horizon recovers most failures, though at matched effective depth part of the tied model's advantage is regularisation. On ARC-AGI the measured contribution of in-step recursion is small, the score falls to zero without the task identifier, and the evidence fits regularisation at least as well as depth. Test-time gains from stochastic restarts are selection gains whose size, on the benchmarks where both are reported, tracks how well the model's halting head verifies an answer. Parameter count is the wrong denominator for all of it. The paper consolidates the measured values in a table and a figure, states a crediting procedure as an algorithm, and specifies the matched comparisons that would assign what remains.

Keywords: recursive reasoning models; tiny recursive model; hierarchical reasoning model; latent recursion; test-time compute; ARC-AGI; Sudoku; evaluation methodology

SONYTECH ORCID: 0009-0003-7224-029X

1. Introduction

In mid-2025 a 27-million-parameter network trained from scratch on about a thousand examples was reported to solve hard Sudoku puzzles and 30x30 mazes on which chain-of-thought language models scored zero, and to reach 40.3 percent on the ARC-AGI-1 evaluation set, above the 34.5 percent then listed for o3-mini-high (Wang et al., 2025). Its authors attributed the result to a brain-inspired architecture: two recurrent modules running at different timescales, a slow one that plans and a fast one that computes, giving "significant computational depth" without backpropagation through time. Three months later single two-layer networks of 5 and 7 million parameters, with no hierarchy and no biological argument, were reported to do better on every one of those benchmarks - 87.4 against 55.0 percent on Sudoku-Extreme for the 5-million-parameter variant without self-attention (the 7-million-parameter attention variant scores 74.7 there), and 85.3 against 74.5 percent on Maze-Hard and 44.6 against 40.3 percent on ARC-AGI-1 for the attention variant - and the paper's title put the lesson plainly: less is more (Jolicoeur-Martineau, 2025). The second model won the first-place paper award of the 2025 ARC Prize (Chollet et al., 2026). A literature followed that treats these Tiny Recursive

Models as a reusable reasoning device: they have been extended with state-space operators (Wang and Reid, 2026), quantised (Ingolfsson et al., 2026), carried into vehicle routing (Bickov and Yorke-Smith, 2026), built on top of a pretrained language model (Manzari et al., 2026), and recast as approximate inference over latent trajectories (Corbett et al., 2026).

The explanation that travels with these results is that recursion does the work. The anchor of this paper, a 2026 workshop paper that improves the models at inference time, states the view compactly: tiny networks "can be surprisingly powerful on structured reasoning tasks", and "the trick is to model reasoning trajectories with a latent dynamical system" (Corbett et al., 2026). On this reading a small network iterated many times substitutes for a large network run once, and the puzzle scores measure what iterated latent computation can do.

A second body of work, nearly all of it from the year after the original models appeared, contests that reading from several directions at once. An independent analysis reported in the TRM paper found that the supervision scheme, not the recursion within each step, doubled ARC accuracy (Jolicoeur-Martineau, 2025). A study of the public ARC verification checkpoint found that replacing each puzzle's identifier with a blank or random token drops accuracy from 40 percent to zero, that a thousand-sample augmentation vote adds about 11 points over a

single pass, and that the first of four recursion steps already delivers 94 percent of the final accuracy (Roya-Azar et al., 2025). A controlled study across six domains found that an optimisation recipe, not an architecture, accounts for much of the variation between the published recursive models, and that the explicit hierarchy is not needed (Shakhvalieva et al., 2026). A mechanistic study argued that the hierarchical model is "guessing" among fixed points rather than reasoning, and lifted its Sudoku accuracy from 55.0 to 96.9 percent with augmentation and ensembling alone (Ren and Liu, 2026). A study of code generation found that a recursive model's advantage over a larger model of the same effective depth is resistance to overfitting, "not greater capability" (Sirivella et al., 2026).

Each side cites real measurements, and the papers on each side rarely report the controls on which the other side's argument turns. In this analysis the author reviews and synthesises published results to show that the conflict is largely an artefact of two confluences. The first is in the word. "Recursion" in these papers names at least three different functions: iterated computation, which lets a fixed network take more serial steps; parameter sharing, which makes a deep computation cheap in parameters and may regularise a model trained on a thousand examples; and a generator of candidate trajectories that a learned halting head can select among at test time. A reported puzzle score also carries ingredients that are not recursion at all: a learned embedding per task, training on the demonstration pairs of the evaluation tasks, heavy augmentation with majority voting, and an optimisation recipe that differs between every pair of published models. The second confluence is between tasks. The evidence that iteration is doing computational work is strong and comes almost entirely from Sudoku and related constraint-satisfaction puzzles. The evidence that recursion contributes little comes almost entirely from ARC-AGI. Read with the task held fixed, the two literatures mostly agree.

The paper argues four claims. First, on Sudoku-Extreme, iteration earns real credit: one-pass Transformers and recurrent networks of matched size solve almost none of the puzzles that weight-tied iterative models solve, and continuing the recursion beyond its training horizon recovers about three quarters of a deterministic model's failures. Second, on ARC-AGI the published measurements put most of the score elsewhere - in task-identity conditioning, in what is in effect test-time training on the evaluation tasks, and in voting - and the one compute-matched comparison that favours weight-tied iteration on ARC does not separate depth from regularisation. Third, the large test-time gains of 2026 are selection gains: they appear where the model's own halting head is a good verifier of its answer, as on Sudoku, and vanish where it is not, as on Maze-Hard and at the ARC-AGI-2 competition metric. Fourth, parameter count is the wrong denominator. The models are tiny in parameters, not in training compute, in per-step inference cost, or in the number of augmented forward

passes behind a single reported answer, and a comparison to language models evaluated zero-shot is a comparison of protocols as much as of architectures.

None of this makes the original results wrong about what they measured. It makes "tiny networks reason through recursion" one of several readings each result supports, and the paper's contribution is to say which reading each result licenses and which comparisons would decide among the rest. Section 2 fixes terms. Section 3 sets out the original claims and the protocols that produced them. Section 4 collects the evidence that iteration does computational work. Section 5 collects the evidence that the ARC score sits elsewhere. Section 6 analyses the test-time results as generate-and-select procedures. Section 7 treats optimisation and architecture as confounds. Section 8 takes up the denominator. Section 9 consolidates the measured values in Table 1 and Figure 1, Section 10 states the crediting procedure as Algorithm 1, and Section 11 specifies the studies that would settle the open part. Section 12 states the best objections to this paper, Section 13 lists what remains unknown, Section 14 its limitations, and Section 15 concludes.

1.1 Relation to the author's earlier papers

Three of the author's published papers bear on this one without overlapping it. One argued that the hallucination detectors that work are external to the model they check (Mahendrakar, 2026a); Section 6 below finds a case where a model's own head does verify its answers well, and argues that the exception is explained by how cheaply the puzzle can be checked, which is consistent with that paper's account rather than a counterexample to it. A second argued that the pass@k crossover between a reinforcement-learned model and its base does not by itself carry the claims made from it (Mahendrakar, 2026b); Section 6 uses a related distinction between coverage, the chance that some sample is right, and selection, the chance that the returned answer is right. A third argued that a scaffold effect is a contrast whose meaning depends on what was held fixed (Mahendrakar, 2026c); Sections 7 and 11 apply the same discipline to architecture comparisons in which the recipe changed along with the architecture. The question here - what a recursive model's puzzle score credits to recursion - is not addressed by any of the three.

2. Terms

2.1 The models

The Hierarchical Reasoning Model (HRM) maps a puzzle, flattened to a token sequence, to an answer of the same shape. It keeps two latent states and updates them with two four-layer Transformer modules: a low-level module updated several times for each update of a high-level module (Wang et al., 2025). The Tiny Recursive Model (TRM) keeps an answer state y and a latent state z and updates both with one two-layer network: z is recomputed n times from the input, y and itself, and

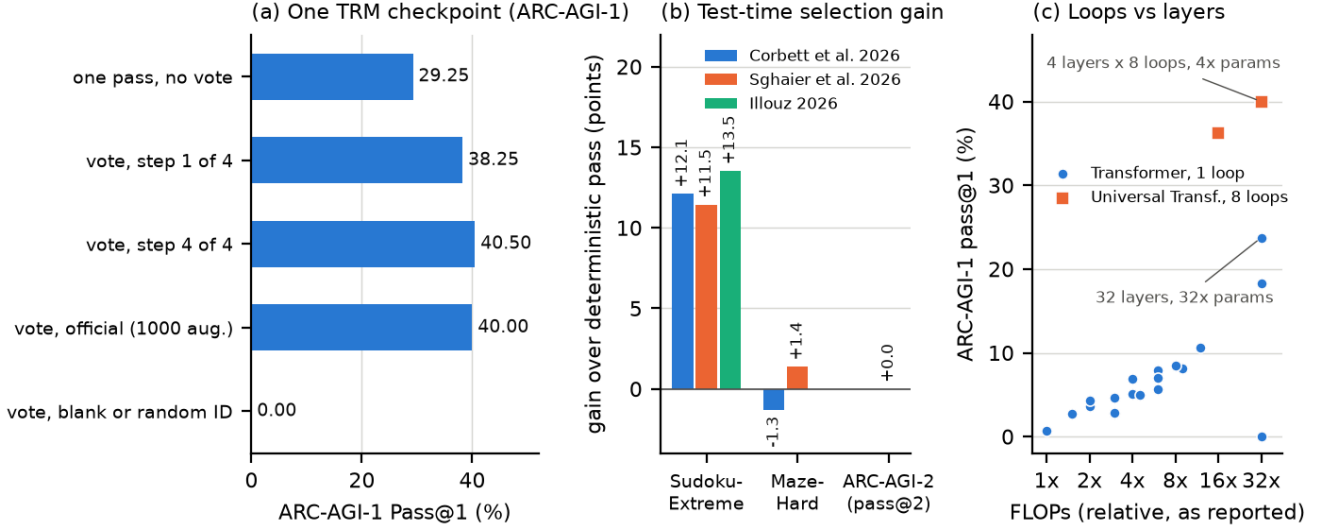


Figure 1. (a) One public TRM checkpoint on the 400 ARC-AGI-1 evaluation tasks under five protocols, Pass@1, from Roye-Azar et al. (2025), Tables 1-3: a single unaugmented pass, the 1000-augmentation vote read out after the first and the fourth of four recursion steps, the official vote, and the vote with the puzzle identifier replaced. (b) Gain of halting-head-selected test-time procedures over the same deterministic checkpoint, in percentage points of exact accuracy, as reported by Corbett et al. (2026, Table 1, guided selection), Sghaier et al. (2026, Section 5.3, best-Q selection; ARC-AGI-2 at pass@2) and Illouz (2026, Tables 1-2, 64 restarts); empty slots were not reported. (c) ARC-AGI-1 pass@1 against FLOPs relative to the smallest model, for vanilla Transformers (one loop) and Universal Transformers (eight loops), from Gao et al. (2025), Table 2.

then y is updated once from y and z (Jolicoeur-Martineau, 2025). The Universal Reasoning Model (URM) is a Universal Transformer variant with short convolutions and truncated backpropagation through its loops (Gao et al., 2025). All three descend from older work on weight-tied iteration: the Universal Transformer, which shares parameters across depth and adds adaptive halting (Dehghani et al., 2018); Adaptive Computation Time and PonderNet, which learn how many steps to take (Graves, 2016; Banino et al., 2021); deep equilibrium models, which solve for a fixed point of a weight-tied layer and differentiate through it implicitly (Bai et al., 2019); and recurrent networks trained to extrapolate from easy to hard problems by iterating longer (Schwarzschild et al., 2021; Bansal et al., 2022). On Sudoku specifically, recurrent relational networks solved 96.6 percent of the hardest puzzles in their test set in 2018 (Palm et al., 2017), and a differentiable satisfiability layer learned to play 9x9 Sudoku from examples (Wang et al., 2019). Iterated neural computation solving Sudoku is not new; what the 2025 models added was the small-data regime and the ARC results.

2.2 What "recursion" names

The paper distinguishes five components that the literature groups under one word.

Inner recursion. The repeated application of the network within one supervision step: HRM’s nested low- and high-level updates, TRM’s n updates of z followed by one update of y , repeated T times.

Outer refinement (deep supervision). The training scheme in which the model runs up to N_{sup} supervision steps on the same example, receives a loss on its answer

after each, and carries its latent state, detached from the gradient, into the next. TRM uses up to 16 such steps (Jolicoeur-Martineau, 2025). Each step is a full forward pass; the latent state carried between them is what makes the model a refinement process rather than an ensemble of independent passes.

Depth extension at test time. Running inner recursion or outer refinement for more steps at inference than were used in training.

Width at test time. Running several trajectories from perturbed states, inputs or initialisations and choosing among the answers.

The halting head. A small output trained, in HRM and TRM, to predict whether the current answer is correct; it decides when to stop during training and is used by the 2026 test-time methods to choose among trajectories (Corbett et al., 2026; Sghaier et al., 2026; Illouz, 2026).

2.3 Three functions recursion could serve

Iteration could matter for at least three different reasons, and each predicts a different signature.

Computation. Iteration lets a fixed network take more serial steps, so problems that need many steps of constraint propagation become reachable. This predicts that one-pass models of the same parameter count fail, that accuracy rises with steps, and that more steps at test time help on harder instances. It does not predict an advantage over an untied network of the same effective depth.

Regularisation. Weight-tying gives a deep computation with few parameters. With a thousand training examples, a model with fewer free parameters may simply overfit less. This predicts an advantage over a larger untied

model of the same effective depth that grows as training data shrink, and no particular dependence on test-time steps.

Generation for selection. A deterministic iterated map started from different states can land in different attractors, so the iteration becomes a sampler of candidate answers; a selector then picks one. This predicts gains from width that are bounded by the selector's accuracy, not by the iteration.

The three are not exclusive, and a single model can benefit from all of them. The point of separating them is that a single score cannot say which one it reflects.

2.4 Ingredients that are not recursion

Identity conditioning. HRM and TRM prepend to each ARC example a learned embedding specific to the puzzle it comes from (Wang et al., 2025; Jolicoeur-Martineau, 2025).

Training on evaluation tasks. For ARC, HRM's training set includes "all demonstration pairs along with test inputs from the evaluation set" (Wang et al., 2025). The test outputs are withheld; the model is trained on the evaluation tasks' worked examples before it is asked their questions. Deliege and Van Droogenbroeck (2026) call this protocol transductive, in the sense that the training process can already adapt to the future test tasks, and contrast it with an inductive protocol in which each test task is first seen at test time.

The word transduction is used in a second sense in the ARC literature, for a model that predicts the test output directly rather than first inferring a program (Li et al., 2024). Both senses apply to these models, and this paper uses the word only in the first.

Augmentation and voting. ARC training uses 1000 augmentations per example (colour permutations, dihedral transformations, translations), and at test time the model is run on 1000 augmented versions of each test input, the predictions are mapped back, and the two most common answers are submitted (Wang et al., 2025; Jolicoeur-Martineau, 2025).

The optimisation recipe. Gradient horizon, the one-step gradient approximation, exponential moving averages of the weights, batch size, learning rates and normalisation all differ across HRM, TRM and URM (Shakhvalieva et al., 2026).

3. The original claims and the protocols behind them

3.1 What was reported

HRM's abstract reports near-perfect performance on "complex Sudoku puzzles and optimal path finding in large mazes" and that HRM "outperforms much larger models with significantly longer context windows" on ARC (Wang et al., 2025). The near-perfect Sudoku figure is on Sudoku-Extreme Full, a set of 3,831,994 training puzzles; the main 1000-example Sudoku-Extreme result is 55.0 percent. The data are split so that "the test set

puzzles cannot be derived through equivalent transformations of any training samples", which is a careful split, and the test puzzles are drawn from the same pool as the training puzzles. HRM's paper also reports that on Sudoku-Extreme Full, widening a Transformer brings no gain while deepening it is critical, and that a standard eight-layer Transformer of HRM's size solves none of the 1000-example Sudoku puzzles but reaches 16.9 percent when trained on the full set.

TRM's paper reports the four improvements already listed and an ablation on Sudoku-Extreme (Jolicoeur-Martineau, 2025). Its Table 1 is the most informative single table in this literature and is revisited in Section 7. Its own conclusion is unusually frank about the open question: "the question of why recursion helps so much compared to using a larger and deeper network remains to be explained; we suspect it has to do with overfitting, but we have no theory to back this explanation."

3.2 The language-model comparison

Both papers set their results beside chain-of-thought language models: DeepSeek R1, Claude 3.7 and o3-mini-high score 0.0 on Sudoku-Extreme and Maze-Hard in the tables, and these three with Gemini 2.5 Pro score between 15.8 and 37.0 percent on ARC-AGI-1 with two attempts; Grok-4-thinking reaches 66.7 percent (Jolicoeur-Martineau, 2025, Tables 4 and 5, some of the language-model rows taken from Wang et al., 2025). These are different protocols, not only different architectures. The recursive models were trained on a thousand puzzles from the test distribution, or on the evaluation tasks' own demonstrations; the language models were evaluated without any training on the task. The comparison shows that a small model trained for a task can beat a large model that was not. It does not show that the small model reasons more effectively per unit of anything, and in particular it does not hold fixed the information available about the test distribution.

The same mismatch recurs in the 2026 test-time work. On a 300-puzzle "golden set" from Pencil Puzzle Bench (Wagh, 2026), Probabilistic TRM (PTRM) reports 91.2 percent against 34.7 percent for the strongest frontier model asked directly and 55.1 percent for an any-of-seven ensemble credited with a perfect verifier, at "less than 0.0001x the cost" (Sghaier et al., 2026). The comparison is restricted to five puzzle types, and the TRM checkpoint was trained on a training split of those same types; the language models were not. The cost figures are inference costs; training the checkpoint on those puzzle types is not part of the comparison. The per-type percentages the paper reports for the language models are multiples consistent with 49 puzzles in total (for example 34.7 percent is 17 of 49); that is the author's inference from the reported figures, and if it is right the headline comparison rests on 49 puzzles.

3.3 What a reported ARC number includes

Putting the components of Section 2.4 together, a reported ARC-AGI-1 accuracy of 44.6 percent for TRM is the output of the following pipeline: a model trained for

about three days on four H100 GPUs (Jolicoeur-Martineau, 2025) on the training tasks, 160 ConceptARC tasks and the evaluation tasks' demonstrations, each example augmented a thousand times and tagged with a learned per-puzzle embedding; then, for each test input, a thousand augmented forward passes of up to 16 supervision steps each, mapped back and voted, with two answers submitted. Every one of those choices is legitimate under the benchmark's public-evaluation rules. None of them is recursion, and a claim about what recursion achieves on ARC has to say how much of the 44.6 points it leaves for recursion to explain.

4. The case for iteration: Sudoku and its relatives

This section states the strongest evidence that iterated computation, not something else, is doing work. Almost all of it comes from Sudoku-Extreme and other constraint-satisfaction or algorithmic tasks.

4.1 One pass fails where iteration succeeds

The cleanest comparison holds parameter count roughly fixed and varies only whether the network refines a problem state. Rodrigues and Kang (2026) train four baselines with HRM's data and procedure, all between 25 and 28 million parameters, on Sudoku-Extreme. The recurrent Transformer and the Universal Transformer reach puzzle accuracies of 51.4 and 48.4 percent at their best checkpoints; the plain eight-layer Transformer never exceeds 0.4 percent and a GRU over the input sequence solves none. The single-state recurrent Transformers are "comparable to HRM", which removes the hierarchy as the explanation while keeping iteration. Huang et al. (2026) report the same contrast with feedforward models of 4, 16 and 64 layers, which solve 1.8, 2.1 and 2.6 percent of Sudoku-Extreme and none of their uniquely solvable maze set, against 84.8 and 44.9 percent for their TRM run and 99.8 and 93.0 percent for their own equilibrium model after test-time depth scaling and 128 restarts; a 256-layer feedforward model ran out of memory.

The Rodrigues and Kang baselines are matched in parameters, not in effective depth, and the comparison therefore shows that iteration with few parameters beats one pass with the same parameters, which is what both the computation and the regularisation accounts predict. Huang et al.'s 4-to-64-layer series is matched in neither: the paper gives no parameter counts for it, and going from 4 to 64 untied layers moves Sudoku accuracy only from 1.8 to 2.6 percent. The same paper does run the comparison that separates the two accounts. On Sudoku-Extreme it sets a 42-block untied feedforward model against a 2-block model iterated 21 times, at the same 42 layer evaluations: the untied model has 105.6 million parameters and the tied one 5.03 million, training accuracy is 93.8 against 94.5 percent, and test accuracy is 2.6 against 32.6 percent (Huang et al., 2026, Table 2). At its best evaluation checkpoint the untied model's training error is 6.2 percent against 3.8 for the 21-

iteration tied model (their Table 6). The authors conclude that "under the matched layer-evaluation budget, the feedforward model underperforms in both training and evaluation, whereas replacing independent layers with repeated applications of shared parameters yields better generalization and a smaller train-evaluation mismatch", and report that a learning-rate sweep does not close the gap: "the feedforward models can fit the training set, but evaluation accuracy remains extremely low". Two models that fit the training puzzles about equally well and differ by 30 points on test puzzles, with 21 times fewer parameters on the better side, is the regularisation signature of Section 2.3 at matched effective depth. It does not exclude computation - the tied model also trains slightly better - but on Sudoku part of what weight-tied iteration earns at this depth is generalisation rather than reach.

4.2 More steps at test time recover failures

The computation account predicts that a model stopped early by its training horizon should improve if allowed to continue. On Sudoku-Extreme this is what happens. In a controlled harness, continuing a deterministic TRM's recursion past its training horizon, with no perturbation at all, recovers 73.8 percent of its standard-pass failures after 16 additional blocks of supervision steps; the same procedure recovers 15.6 percent on Pencil Puzzle Bench validation (Illouz, 2026). Illouz concludes that "on Sudoku-Extreme, most failures thus come from trajectories that the horizon cuts short." Huang et al. (2026) report that a model trained with 16 iterations keeps improving to over 1,024 iterations at test time, which they describe as equivalent to more than 40,000 effective layers, and its best Sudoku-Extreme result, 99.8 percent, combines 64 test-time iterations with 128 restarts from random initial states. A model trained with loss only on its final step and detached warm-up iterations reports test-time scaling to about 20,000 refinement steps against roughly 20 in training, reaching 97.5 percent on Sudoku-Extreme (Hakimi, 2026).

The pattern matches older results on algorithmic extrapolation. When a recurrent network learns prefix sums, mazes or chess puzzles on small instances with a short iteration budget, letting it iterate longer at test time lets it handle bigger and harder instances (Schwarzschild et al., 2021), provided that a copy of the input is kept available at every step and training discourages iteration-specific behaviour (Bansal et al., 2022). A fixed-point analysis of looped Transformers on chess, Sudoku and prefix sums proves that without such recall a looped network has only countably many fixed points and cannot be strongly input-dependent, and finds that recall combined with outer normalisation produces reachable, input-smooth fixed points (Labovich, 2026).

4.3 Iteration can match depth

The computation account would be strongest if iterating a k -layer block L times did as well as a kL -layer untied network. For synthetic reasoning problems such as addition and p -hop induction, a k -layer Transformer

looped L times "nearly matches the performance of a kL -layer non-looped model" and is much better than a k -layer model (Saunshi et al., 2025). Looped Transformers with an adaptive number of steps also generalise in length on tasks with a known iterative solution (Fan et al., 2024), and a looped Transformer of fixed size can in principle emulate an instruction-set computer (Giannou et al., 2023). At language-model scale, a 3.5-billion-parameter recurrent-depth model improves on reasoning benchmarks as its recurrence is unrolled, up to a computational load equivalent to 50 billion parameters (Geiping et al., 2025). None of these is a puzzle benchmark of the kind the tiny models report, but they establish that iterated depth can carry computation that would otherwise need more layers.

4.4 A compute-matched comparison on ARC

The one result in this literature that matches FLOPs while varying weight-tying, on ARC itself, favours iteration. Gao et al. (2025, Table 2) train vanilla Transformers of 2 to 64 layers and widths of 256 to 1024 alongside Universal Transformers with 8 loops on ARC-AGI-1 under the HRM/TRM data setting. At 32 times the FLOPs of the smallest model, the best vanilla model - 32 layers at width 512, with 32 times the parameters - reaches 23.75 percent pass@1; a Universal Transformer of 4 layers looped 8 times, with 4 times the parameters, reaches 40.00 percent. Figure 1(c) plots the full table. No vanilla model in the table exceeds 23.75 percent.

This is the strongest evidence against the deflationary reading of ARC. It is also, read carefully, evidence that does not choose between computation and regularisation. The two models in the comparison have the same FLOPs and differ by a factor of eight in parameters, and Gao et al.'s own reading is that the gains of Universal Transformers on ARC-AGI-1 "arise from substantially higher parameter efficiency rather than increased model scale or computational budget". That excludes a larger FLOP budget, not computation: the same passage goes on to credit iterated computation, citing earlier analyses that many reasoning tasks gain more "from iterative computation than from increasing the number of independent layers" and concluding that "recurrent computation converts the same budget into increased effective depth" (Gao et al., 2025). Section 5.4 returns to this.

4.5 What Section 4 establishes

On Sudoku-Extreme, the record supports a specific claim: weight-tied iteration with few parameters solves puzzles that one-pass models of the same parameter count do not, accuracy rises with iterations, and continuing the recursion recovers most failures. The claim is about iteration, not about hierarchy - the hierarchical model is matched by single-state recurrent Transformers (Rodrigues and Kang, 2026) and beaten by a single shared network (Jolicoeur-Martineau, 2025). It is also about an in-distribution test: TRM trains on 1000 Sudoku-Extreme puzzles and tests on 423,000 from the same distribution, so the results show iteration helping

within a distribution, not easy-to-hard extrapolation. The out-of-distribution evidence specific to the 2025-2026 models is thinner but positive: on an arithmetic out-of-distribution split, a retrained TRM scores 36.2 percent against 7.5 for a non-recurrent dense control, and a recursive model with a stabilised training recipe raises that to 71.2 percent, a further gain that is a recipe effect rather than an iteration effect (Shakhvalieva et al., 2026); and a 0.53-million-parameter recursive routing model trained on 100-customer problems transfers zero-shot to 500 and 1000 customers with optimality gaps of 15.6 and 21.7 percent under 256-candidate search (Bickov and Yorke-Smith, 2026).

5. The case against crediting recursion on ARC

This section collects the measurements that locate the ARC score in components other than iterated computation. They come from four independent groups, use the public checkpoints or the released code, and agree with each other more than with the original framing.

5.1 The score depends entirely on knowing which task it is

Roye-Azar et al. (2025) evaluate the ARC Prize verification checkpoint of TRM on the 400 public ARC-AGI-1 evaluation tasks. With the correct puzzle identifiers the full pipeline scores 40.00 percent Pass@1. Replacing every identifier with a single trained "blank" identifier, or each with a random identifier from another task, leaves the input grids, the augmentations and the weights unchanged, and drops accuracy to 0.00 percent in both conditions (their Table 2; Figure 1(a)). This checkpoint does not solve any ARC evaluation task from its input grids alone. It solves them only when told, through an embedding learned during training on that task's demonstrations, which task it is looking at.

That embedding was learned on the evaluation task's own demonstration pairs (Section 2.4). The ARC Prize Foundation's report on the 2025 competition draws the consequence directly. It places HRM and TRM among "zero-pretraining" methods that fit a network to encode a task-solving program "using only examples of that task", and observes of this family that "all material task-specific training occurs at test time" (Chollet et al., 2026). On this description the ARC result of a tiny recursive model is a test-time training result. That puts it in a different comparison class from the language models in Section 3.2, and in the same class as the methods that won the ARC Prize: test-time training produced the top Kaggle scores in both 2024 and 2025 (Chollet et al., 2026), after a 2024 competition in which it and deep-learning-guided program synthesis raised the private-set state of the art from 33 to 55.5 percent (Chollet et al., 2024), an 8-billion-parameter language model with test-time training on in-context examples reached 53.0 percent on the public ARC validation set (Akyurek et al., 2024), a Vision Transformer trained from scratch on ARC data and adapted per task reports 54.5 percent on ARC-1, and

60.4 percent when ensembled with a U-Net (Hu et al., 2025), and a 76,000-parameter model with no pretraining at all, trained only on the target puzzle by minimising description length, solves 20 percent of the ARC-AGI-1 evaluation set (Liao and Gu, 2025).

Two controlled studies confirm that the protocol, not the architecture alone, sets the number. Deliege and Van Droogenbroeck (2026) re-run TRM and the Vision ARC model under a common inductive protocol - pretrain on the training tasks, then train separately on each test task - crossing five pretraining budgets with four test-time training budgets. TRM's best pass@2 rises from 19.63 percent without pretraining to 37.50 percent with it, against the 44.6 percent reported under the original transductive protocol, and raising the number of test-time sample presentations improves pass@2 in every comparison, by 7.99 and 17.76 points for TRM without pretraining. They report one run per configuration and say so. McGovern (2025) shows the same dependence from the competition side: the original TRM pipeline required "a level of compute far in excess of what is allowed during the competition"; a model pretrained for 48 hours on four H100 GPUs on 1,280 public tasks and then fine-tuned for 12,500 steps inside the competition environment reached 6.67 percent on the semi-private ARC-AGI-2 tasks, and full fine-tuning of the network did better than fine-tuning the task embeddings alone or a low-rank adapter. Separately, optimising only a task embedding - under 0.01 percent of a Vision ARC model's parameters - already solves a non-trivial fraction of ARC-AGI-1 tasks, and the embedding learned that way supports interpolation between rules but not extrapolation beyond them (Deliege et al., 2026).

5.2 A quarter of the score is the vote

On the same checkpoint, the official pipeline with a thousand augmentations and majority voting scores 40.00 percent Pass@1; a single forward pass on the canonical, unaugmented input scores 29.25 percent (Roye-Azar et al., 2025, Table 1). The vote supplies 10.75 points, about 27 percent of the reported score. Their early-training experiments suggest that heavy augmentation first broadens the distribution of candidate answers and improves multi-sample success before it improves single-pass accuracy. The mechanism is the one that self-consistency exploits in language models - sample diverse paths and return the most frequent answer (Wang et al., 2022) - and the one that a strong open ARC system built on a language model uses explicitly, with task-specific augmentations at training, generation and scoring and the model's own probabilities as the selector, reaching 71.6 percent on the public evaluation set (Franzen et al., 2025). On ARC, then, part of what is reported as a tiny model's accuracy is the accuracy of an ensemble of a thousand of its forward passes over transformed inputs.

5.3 Effective recursion on ARC is shallow

If iterated computation carried the ARC score, accuracy should build over recursion steps. On the verification checkpoint it does not. With the full voting pipeline applied at every depth, Pass@1 is 38.25 percent after the first of four recursion steps, 40.38 after the second, 40.13 after the third and 40.50 after the fourth, and iterating to six steps leaves it at 40.50 (Roye-Azar et al., 2025, Table 3). The first step delivers 94.4 percent of the final accuracy. A separate line of work argues that not every recursive step contributes effective depth, calls the remainder dead compute, and reports keeping TRM's performance with 18 times fewer forward passes after recasting the recursion as policy improvement (Asadulaev et al., 2025). Shakhvalieva et al. (2026) find the same contrast across tasks under their unified pipeline: raising the adaptive-computation budget beyond its training value gives the largest gains on arithmetic and Sudoku and limited gains on Game of Life, mazes and ARC-AGI.

The ablation reported in the TRM paper points the same way at the level of training. Citing an independent analysis by the ARC Prize Foundation, Jolicoeur-Martineau (2025) reports that deep supervision doubled ARC accuracy over single-step supervision, from 19 to 39 percent, while HRM's hierarchical recursion within each supervision step improved on a model with a single forward pass only from 35.7 to 39.0 percent. Ge et al. (2025) remove HRM's high-level module and expand the low-level module to eight layers, eliminating the separate hierarchical state; in the configuration their figure caption describes as equivalent to an eight-layer plain Transformer, run inside HRM's deep-supervision loop, the result performs similarly to or slightly better than the original HRM and trains faster (1 h 48 min against 4 h 21 min on one A100), and the authors relate HRM's training to latent consistency models, noting that a well-trained HRM on Sudoku needs as few as 2 to 4 inference steps. The paper does not name the task behind that comparison; its other experiments are on Sudoku, and the reported runtime is short for an ARC run, so the result bears on the hierarchy and is not counted here as ARC evidence. The paper's v2 states that errors were found in some v1 results and removed; the findings used here are from v2.

Taken together, these results say something narrower than "recursion does not matter". They say that on ARC, the iteration that matters is the outer refinement loop with its per-step supervision, and that the inner recursion within each step, the part the original papers presented as the reasoning, contributes a few points.

5.4 Reconciling the ARC evidence with the compute-matched result

Section 4.4 reported the one compute-matched ARC comparison: a 4-layer Universal Transformer looped 8 times reaches 40.00 percent pass@1 where a 32-layer vanilla Transformer of equal FLOPs reaches 23.75 (Gao et al., 2025). Sections 5.1 to 5.3 report that the inner

recursion contributes little on ARC. These can both be true, and the regularisation account of Section 2.3 is the simplest way they could be.

The matched-FLOPs comparison is unmatched in parameters by a factor of eight. Its authors read the gain as effective depth bought by parameter sharing ("recurrent computation converts the same budget into increased effective depth"; Gao et al., 2025), but the design does not separate that from a regularisation effect of having eight times fewer parameters. Three other observations point toward regularisation as at least part of the ARC story. First, adding capacity hurts these models: TRM's author reports that replacing its MLPs with mixture-of-experts layers decreased generalisation "massively" and that such layers "add too much unnecessary capacity, just like increasing the number of layers does" (Jolicoeur-Martineau, 2025), and Shakhvalieva et al. (2026) find that eight-layer blocks, wider states and more heads bring no consistent gain. Second, the two studies that compare a recursive model with an untied model of the same effective depth while tracking training accuracy both locate the advantage in generalisation. On Sudoku-Extreme, an untied 42-block model with 105.6 million parameters fits the training puzzles about as well as a 5-million-parameter tied model at the same 42 layer evaluations and scores 2.6 against 32.6 percent on test puzzles (Huang et al., 2026; Section 4.1). On code generation, the untied model "fits and generates better at its validation optimum" and the recursive model's advantage lies in "resistance to overfitting, not greater capability" (Sirivella et al., 2026). Neither is on ARC, so for ARC both are indirect evidence. Third, TRM's author suspects overfitting as the explanation for why recursion beats a larger, deeper network.

None of this shows that iteration contributes nothing on ARC. The outer loop's per-step supervision is itself iteration, and the 19-to-39 result credits it heavily. What the record does not contain is the factorial comparison that would separate the three functions on ARC: the same effective depth with and without weight-tying, at matched parameters and matched FLOPs, under an inductive protocol without task identifiers. Section 11 specifies it. Until it is run, "recursion" on ARC is credited with an effect that the evidence fits at least as well as an account in which a few-parameter model trained on a few demonstrations overfits less, is refined by a supervised outer loop, and is helped by a learned task embedding and a thousand-way vote.

6. Test-time gains are selection gains

The 2026 test-time methods - the anchor paper among them - are the most direct evidence offered that recursive models leave reasoning capacity untapped. This section argues that what they demonstrate is a generate-and-select procedure whose payoff is set by the quality of the selector.

6.1 Five methods, one structure

Every method in this group perturbs the deterministic recursion to produce several trajectories and uses a signal internal to the model to choose among them. The anchor runs a particle filter: 16 particles, each perturbed by Gaussian noise during the recursion, reweighted at each of 48 outer steps by the halting head, with the weighted most-likely answer returned (Corbett et al., 2026). PTRM injects noise at each deep recursion step, runs up to 100 rollouts, and returns the rollout whose halting-head value is highest (Sghaier et al., 2026). Gated Answer Resampling restarts only the puzzles whose halting head rejects the first answer, resampling the answer state y and keeping the first answer the head accepts (Illouz, 2026). Halt-first selection runs an ensemble of TRMs and returns the answer of whichever halts first (Dillon, 2026). Augmented HRM perturbs the input by relabelling tokens, ensembles checkpoints from one training run, and votes among the passes that halt, and it also retrains the model on data mixed with simplified copies of the training puzzles, a training-time change that alone moves HRM from 55.0 to 59.9 percent (Ren and Liu, 2026, Table 1). Equilibrium Reasoners scale both depth and breadth from random initial states with convergence-based selection (Huang et al., 2026), and two further lines train the noise into the model rather than adding it at inference (Baek et al., 2026) or add a learned reweighting component to TRM's inference scheme (Komisarczyk et al., 2026).

6.2 Where the selector is good, the gain is large

On Sudoku-Extreme the gains are large and consistent. The anchor takes TRM from 85.9 to 98.0 percent exact-solve accuracy on 10,000 test puzzles without retraining, with hyperparameters chosen on a disjoint 500-puzzle validation subset and results averaged over five folds and five seeds (Corbett et al., 2026). PTRM takes its baseline from 87.3 to 98.75 percent; GAR from 85.90 to 99.43 percent at 2.03 times the cost of a standard pass; halt-first selection reaches 97 percent against 91 percent for probability averaging and 85.5 percent for a single model; Augmented HRM moves HRM from 55.0 to 96.9 percent (Sghaier et al., 2026; Illouz, 2026; Dillon, 2026; Ren and Liu, 2026).

The anchor's own table shows where the gain comes from. Perturbing the recursion without guidance lowers accuracy slightly, from 85.9 to 84.3 percent; the best particle in each cloud, chosen with knowledge of the answer, would score 98.0; and guidance by the halting head scores 98.0, closing the whole gap to that oracle (Corbett et al., 2026, Table 1). On the 1,400 puzzles the deterministic model fails, unguided noise solves 35.7 percent and guided selection 85.9. The exploration supplies coverage; the halting head converts coverage into accuracy. GAR's analysis says the same from a different angle: on Sudoku-Extreme its head restarts 99.3 percent of the failed puzzles and none of the solved ones (Illouz, 2026). Dillon (2026) reports that 89 percent of a baseline TRM's failures are "selection problems", implying an accuracy ceiling of 99 percent with a perfect

selector. The same holds on Pencil Puzzle Bench validation: GAR's head re-opens 87.5 percent of the failures and none of the successes, and GAR raises accuracy from 68.8 to 85.6 percent at 7.08 times the cost (Illouz, 2026), while PTRM reports for the same benchmark's validation types that its oracle and its halting-head choice differ by less than a point, "making the Q head a strong verifier on this validation set" (Sghaier et al., 2026).

6.3 Where the selector is weak, the gain disappears

On Maze-Hard the same procedures do much less. The anchor's guided selection scores 85.3 percent against a deterministic 86.6, while the best particle would have scored 90.9; its diagnostic flags the halting head as too flat to discriminate, a priori and without labels (Corbett et al., 2026). PTRM finds a correct path among its 100 rollouts 95.63 percent of the time, against an 83.8 percent baseline, but selecting by the halting head returns 85.17 percent and majority voting 86.73 (Sghaier et al., 2026). On ARC-AGI-2, PTRM raises pass@1 from 7.36 to 8.47 percent and pass@100 from 14.31 to 15.97, while "matching it at pass@2", the competition metric. Figure 1(b) collects the gains by benchmark.

The resulting pattern is that test-time gain tracks selector quality, not the amount of recursion. The same structure has been analysed for language models. Coverage from repeated sampling grows roughly log-linearly with the number of samples (Brown et al., 2024), but when the verifier has a non-zero false-positive rate, resampling cannot push accuracy past a bound set by that rate, "regardless of compute budget" (Stroebl et al., 2024). The recursive-model results are a clean instance: on Sudoku the internal verifier has nearly no false positives and the bound is near 100 percent; on Maze-Hard it does not, and most of the coverage gain is lost at selection.

6.4 Why Sudoku has a good selector

A plausible reason the halting head verifies Sudoku well is that Sudoku is cheap to verify. Whether a completed grid satisfies the row, column and box constraints is a local check that a small network can learn, and a correct Sudoku solution is unique. A maze path's optimality is a global property, and the original Maze-Hard data contain instances with several shortest paths of which only one is labelled; Huang et al. (2026) built a uniquely solvable variant for exactly this reason. Pencil Puzzle Bench gives mixed evidence on the local-check reading. The halting head verifies well on its validation set overall (Section 6.2), which includes nurikabe and heyawake, puzzles whose rules require shaded or unshaded regions to be connected across the grid, a global constraint, and it re-opens no solved puzzle of any type. But its recall is lowest on two of those connectivity types: it re-opens 72.7 percent of the failures on heyawake and 66.7 percent on nurikabe, against 100 percent on lightup and 98.2 on sudoku (Illouz, 2026, Appendix G), and GAR's author notes that "the gains are smaller on nurikabe and heyawake". Local checkability is therefore at most part of the explanation. The author suspects that part of the

Maze-Hard selection failure reflects that label ambiguity: a head trained to predict agreement with one labelled path cannot learn to accept an equally short unlabelled one, and an oracle that knows the labelled path is not the right ceiling for a selector that cannot. That is a conjecture; no published study has run the 2026 test-time methods on the uniquely solvable maze set, and Section 11 proposes that it be done.

This reading connects to the author's earlier argument that working hallucination detectors are external to the model they check (Mahendrakar, 2026a). The halting head is internal, and on Sudoku it works. The explanation offered here is that it works where verification is cheap enough to be learned as a side task. On ARC, where checking an answer requires knowing the rule, the selector that does the work at the competition metric is the vote over augmentations, an external consistency check, and the internal head adds nothing at pass@2.

6.5 What the test-time results credit to recursion

GAR's controlled comparisons add one more qualification. At equal compute, resampling the answer state y recovers more failures than any perturbation of the latent reasoning state z , and one step into a continued retry, resampling y has moved z further than resampling z itself, "as if z were largely recomputed from the input and y " (Illouz, 2026). If that holds generally, the latent state that TRM describes as the model's reasoning carries less of the trajectory's identity than the answer state does, and the useful diversity comes from restarting the answer. On this evidence the test-time literature credits recursion with being a good generator of diverse candidate answers on tasks with many attractors, and credits the halting head - or, on ARC, the vote - with the selection. It does not show that the recursion reasons further when given more compute, except in the depth-continuation results of Section 4.2.

7. Optimisation and architecture as confounds

7.1 One table, many levers

TRM's ablation on Sudoku-Extreme shows how much of a recursive model's accuracy moves with choices other than recursion (Jolicoeur-Martineau, 2025, Table 1). From the full model's 87.4 percent, reverting to HRM's one-step gradient approximation drops accuracy to 56.5; removing the exponential moving average of weights drops it to 79.9; using self-attention instead of an MLP over positions drops it to 74.7; using four layers instead of two (with $n=3$) drops it to 79.5; using separate networks for the two states drops it to 82.4; and shortening the recursion to $T=2$, $n=2$, which cuts effective depth per supervision step from 42 to 12, drops it to 73.7. The recursion-depth change costs 13.7 points. The gradient approximation costs 30.9, and the moving average 7.5. The MLP choice is task-specific: it is best on Sudoku and worse on the larger ARC and maze grids, where the self-attention variant is used (Jolicoeur-Martineau, 2025). Several ablated variants also differ in

parameter count (5 to 10 million), which the table reports alongside accuracy.

The most consequential lever in that table is not part of recursion: it is how gradients flow through it. HRM backpropagates through only the last two of six function evaluations, justified by assuming the recursion reaches a fixed point; TRM's author shows that the residuals in HRM's own example remain well above zero when the approximation is applied, and backpropagates through the full recursion of each supervision step instead (Jolicoeur-Martineau, 2025). Ren and Liu (2026) report the same fixed-point violation from the other side: HRM can fail on a puzzle with a single unknown cell because, once at the solution, it does not stay there.

7.2 Controlled comparisons point to the recipe

HRM, TRM and URM change "architecture, gradient horizon, optimizer, regularization, and evaluation setup together", which makes their published differences hard to attribute (Shakhvalieva et al., 2026). Under one pipeline across six algorithmic domains, those authors find that an intermediate gradient horizon, large physical batches rather than gradient accumulation, and controlled updates of the recurrent state are the main ingredients of stable and generalisable training; that shared and separate recurrent modules at matched parameter count trade wins across domains, with shared modules better on mazes and ARC-AGI-1 and separate modules slightly better on Sudoku and Game of Life; and that a stabilised 13.6-million-parameter model reaches 98.41 percent on Sudoku and 59.5 percent pass@2 on ARC-AGI-1. Their summary is that performance "depends strongly on how recurrence is optimized and stabilized". A model that keeps TRM's backbone but trains with detached warm-up iterations and a loss only on the final step reports training more than 20 times faster than TRM with about a fifth of its error rate on Sudoku-Extreme (Hakimi, 2026), which, if it holds up, is evidence that the per-step deep supervision credited with doubling ARC accuracy is not the only way to train these models well, at least on Sudoku.

7.3 Fragility that recursion introduces or amplifies

Some properties of these models are consequences of recursion and cut against treating it as a free lunch. Because the same block is applied many times, a per-tensor 4-bit quantisation of activations lets a systematic bias accumulate across steps and drops Sudoku exact-solution accuracy from 84.1 to 0.0 percent; per-block MXInt4 scaling brings it back to 80.1 percent (Ingolfsson et al., 2026). The failure is not unique to the tied model - the same paper reports that "Full Int4 (4-bit w + a) collapses the tied model to 0.0% and the untied model to 1.6%" - so reuse amplifies a weakness that the untied control shares rather than creating it. The recursion has multiple fixed points, some wrong, in which a deterministic model can be trapped (Ren and Liu, 2026), and the test-time literature of Section 6 exists largely to escape them. Optimisation of these models is described by its most systematic study as "often unstable", and

depends on controlling how the recurrent state is updated (Shakhvalieva et al., 2026). Probe directions that linearly decode task variables from HRM's recurrent states have ablation effects comparable to random directions, and HRM is better described as "constraint-aware iterative refinement on a puzzle-specific solution state" than as a planner and executor (Rodrigues and Kang, 2026). Mechanistic interpretation of what the recursion computes is at an early stage.

8. The denominator

8.1 Parameters are not compute

A parameter count describes stored weights. It does not describe training compute, inference compute, memory during training, or latency, and these cost indicators can disagree and even reverse the ranking of models; parameter sharing across depth is one of the standard cases in which they come apart (Dehghani et al., 2021). Recursive models are the extreme case. TRM's two-layer network is applied 21 times per supervision step, an effective depth of 42 layers, for up to 16 supervision steps per answer during training, and at evaluation on ARC each test input is run in 1000 augmented versions. In the code-generation comparison, an autoregressive TRM costs roughly 175 times as much per step as a parameter-matched control and is "parameter-efficient but not compute-efficient" (Sirivella et al., 2026). The ARC Prize report describes the zero-pretraining family as "extremely small relative to their ARC-AGI performance" (Chollet et al., 2026), which is accurate of parameters and silent about compute.

8.2 Inference cost is not training cost

The "less than 0.0001x the cost" comparison of Section 3.2 counts the cost of an inference attempt (Sghaier et al., 2026). It does not include training a checkpoint on the benchmark's own puzzle types, which the language models in the comparison did not receive. On ARC, inference and training are entangled, because the evaluation tasks are trained on (Section 5.1). The controlled ARC study reports its compute: pretraining the four TRM checkpoints took 408.5 A100 GPU-hours, and its twenty 400-task test-time training runs for TRM took about 1,623.7 GPU-hours (Deliege and Van Droogenbroeck, 2026). By the author's arithmetic from those two figures, that is about 0.2 GPU-hours of test-time training per task per run, before any inference. For scale, the 2025 competition's winning entry reached 24.03 percent on the private ARC-AGI-2 set at a reported \$0.20 per task (Chollet et al., 2026).

8.3 A thousand examples is not small data

HRM and TRM are described as trained on about a thousand examples. Each Sudoku example is augmented 1000 times by shuffles that preserve its validity, and each ARC example 1000 times (Jolicoeur-Martineau, 2025). The training distribution is narrow - one puzzle type, and test puzzles from the same pool - and the comparison models in Section 3.2 had seen no puzzles of

that type under the evaluation protocol. "Small data" is accurate about the number of distinct puzzles and not about the number of training presentations, and it describes an in-distribution regime. A model's ability to learn a narrow, fully specified constraint-satisfaction task from a thousand puzzles is a real result. It is a different result from general reasoning, and the ARC Prize's own framing of the benchmark as a test of generalisation to novel tasks (Chollet, 2019; Chollet et al., 2025) is met by these models only under the test-time training reading of Section 5.1.

9. Consolidated evidence

Table 1 collects the measured values on which the argument rests, one published result per row. The last column is the author's classification of which component of Section 2 the comparison isolates; it is the column a reader should check row by row. "Iteration" means a contrast in which iterated computation varies while parameters are roughly fixed; "outer loop" means deep supervision; "identity", "vote", "selection" and "recipe" refer to the ingredients of Sections 2.2 and 2.4.

Figure 1 puts the three central contrasts side by side. Panel (a) shows on one checkpoint that the vote supplies about a quarter of the ARC score, that with the vote applied the first recursion step already reaches 94 percent of the final score, and that without the task identifier there is nothing. Panel (b) shows that the test-time gains are large on Sudoku-Extreme and near zero or negative on Maze-Hard and at the ARC-AGI-2 competition metric, which is the selection pattern of Section 6. Panel (c) shows the compute-matched comparison that most favours weight-tied iteration on ARC, with the parameter difference stated in the annotations.

10. A procedure for crediting a score

Algorithm 1 states the decision structure the paper argues for: given a reported score from a recursive model, what it licenses about recursion. It is a synthesis of the arguments above, not a procedure any cited study ran.

Applied to TRM on Sudoku-Extreme, S1 finds a training set of 1000 in-distribution puzzles with 1000 validity-preserving shuffles each and no task identifiers or voting at test; S2 has little to remove; S3 classifies Sudoku as locally checkable with a unique solution, so the 2026 test-time gains go to the halting head; S4(c) credits iteration through the depth-continuation result (Illouz, 2026); S4(a) has one comparison at matched effective depth, which favours the tied model with 21 times fewer parameters and almost the same training accuracy, so that its advantage shows up as generalisation (Huang et al., 2026); S5 records 5 million parameters, an effective depth of 42 layers per supervision step, and up to 16 supervision steps. The result: on Sudoku, iteration as computation earns credit through the depth-continuation results, and at matched depth part of the tied model's advantage is regularisation.

Applied to TRM on ARC-AGI-1, S1 finds training on the evaluation tasks' demonstrations, per-task identifiers and a thousand-way vote; S2 removes about 11 points with the vote and the whole score with the identifier; S3 classifies ARC as expensive to verify, consistent with the absence of a pass@2 gain from halting-head selection; S4(a) has one compute-matched comparison that favours tying but is unmatched in parameters, S4(b) has no direct ARC evidence, and S4(c) finds no gain past four steps; S5 records 7 million parameters, roughly three days of four-GPU training that includes the evaluation tasks, and a thousand forward passes per answer. The result: on ARC, the residual is credited to the outer refinement loop and to an unresolved mixture of computation and regularisation, and the score as a whole is a test-time training result.

11. Studies that would settle the open part

Five studies would settle most of what Section 10 leaves open. Each can be run with released code and checkpoints.

Study 1: the three-way factorial. On Sudoku-Extreme and on ARC-AGI-1 under an inductive protocol, cross three factors: weight-tied versus untied blocks; effective depth per supervision step (for example 12 and 42 layers); and number of distinct training puzzles (for example 250, 1000 and 4000). Match FLOPs across the tied and untied arms, and report parameter counts. The computation account predicts that tying does not help at matched effective depth once the untied model is regularised to the same validation loss; the regularisation account predicts that the tied advantage shrinks as training data grow. This is the experiment that would decide Section 5.4.

Study 2: ARC without identifiers. Train TRM on ARC with the per-task embedding removed and the task's demonstrations supplied in the input instead, as the language models in the comparison receive them, and report single-pass and voted accuracy under an inductive protocol. If the score survives, recursion can be credited with in-context rule induction on ARC; if it does not, the ARC result is a test-time training result and should be compared only with test-time training baselines at matched budget (Deliege and Van Droogenbroeck, 2026).

Study 3: selector quality against test-time gain. For each benchmark on which the 2026 test-time methods have been run, report the halting head's discrimination between correct and incorrect final answers (for example its area under the ROC curve on deterministic outputs) alongside the oracle gap and the realised gain. The prediction of Section 6 is that realised gain is a monotone function of discrimination across benchmarks. A benchmark with high discrimination and low gain would refute it.

Study 4: the maze label check. Re-run the anchor's guided selection and PTRM on the uniquely solvable maze set of Huang et al. (2026) alongside Maze-Hard,

Table 1. Published results bearing on what a recursive model's score credits, one row per result. All values are as reported by the source named on the row.

Source	Task	Comparison	Reported result	Isolates
Rodrigues and Kang 2026	Sudoku-Extreme	recurrent vs one-pass, 25-28M	51.4 / 48.4% vs 0.4 / 0.0%	iteration
Huang et al. 2026	Sudoku-Extreme	4-64 layer feedforward vs TRM	1.8-2.6% vs 84.8%	iteration or sharing
Huang et al. 2026	Sudoku-Extreme	42 untied vs 2 tied blocks x 21, equal depth	2.6% vs 32.6% (train 93.8 vs 94.5%)	sharing
Wang et al. 2025	Sudoku-Extreme	8-layer Transformer, 1K vs full data	0.0% vs 16.9%	data
Illouz 2026	Sudoku / PPBench	continue past horizon, 16 blocks	73.8% / 15.6% of failures fixed	depth at test
Huang et al. 2026	Sudoku-Extreme	16 train vs 1,024+ test iterations	over 99% with restarts	depth and width
Gao et al. 2025	ARC-AGI-1	4x8 looped vs 32-layer, equal FLOPs	40.00% vs 23.75% pass@1	iteration or sharing
Jolicoeur-Martineau 2025	ARC-AGI	deep vs single-step supervision	39% vs 19%	outer loop
Jolicoeur-Martineau 2025	ARC-AGI	hierarchical recursion vs one pass	39.0% vs 35.7%	inner recursion
Roye-Azar et al. 2025	ARC-AGI-1	correct vs blank or random ID	40.00% vs 0.00%	identity
Roye-Azar et al. 2025	ARC-AGI-1	1000-aug. vote vs single pass	40.00% vs 29.25%	vote
Roye-Azar et al. 2025	ARC-AGI-1	recursion step 1 vs step 4	38.25% vs 40.50%	inner recursion
Deliege and Van Droogenbroeck 2026	ARC-AGI-1	inductive, no pretrain vs pretrain	19.63% vs 37.50% pass@2	protocol
McGovern 2025	ARC-AGI-2	in-competition fine-tune	6.67% semi-private	protocol
Jolicoeur-Martineau 2025	Sudoku-Extreme	full recursion vs 1-step gradient	87.4% vs 56.5%	recipe
Jolicoeur-Martineau 2025	Sudoku-Extreme	T=3, n=6 vs T=2, n=2	87.4% vs 73.7%	inner recursion
Jolicoeur-Martineau 2025	Sudoku-Extreme	with vs without weight EMA	87.4% vs 79.9%	recipe
Ge et al. 2025	not named	HRM vs 8-layer L-module only	similar or slightly better	hierarchy
Shakhvalieva et al. 2026	6 domains	shared vs separate modules	no consistent winner	hierarchy
Corbett et al. 2026	Sudoku-Extreme	TRM vs unguided vs guided	85.9 / 84.3 / 98.0%	selection
Corbett et al. 2026	Maze-Hard	TRM vs guided vs oracle	86.6 / 85.3 / 90.9%	selection
Sghaier et al. 2026	Maze-Hard	base vs pass@K vs best-Q@K	83.8 / 95.63 / 85.17%	selection
Sghaier et al. 2026	ARC-AGI-2	deterministic vs PTRM, pass@2	unchanged	selection
Illouz 2026	Sudoku-Extreme	TRM vs gated answer resampling	85.90% vs 99.43% at 2.03x	selection
Dillon 2026	Sudoku-Extreme	single vs halt-first ensemble	85.5% vs 97%	selection
Ren and Liu 2026	Sudoku-Extreme	HRM vs relabel + bootstrap + mixing	55.0% vs 96.9%	width and data
Sirivella et al. 2026	code generation	TRM vs untied, equal depth	untied fits and generates better	sharing
Ingolfsson et al. 2026	Sudoku	8-bit vs per-tensor 4-bit	84.1% vs 0.0%	fragility

Algorithm 1. Crediting a recursive model's score

Input: a reported accuracy A on task X for model M S1: Fix the protocol. Record: training data and whether it includes the evaluation tasks' demonstrations; per-task identifiers; augmentation count; voting or selection rule; attempts per answer. If any is present, A is a pipeline score; go to S2 before crediting M. S2: Remove non-recursive ingredients one at a time, using published ablations where they exist (single pass vs vote; true vs blank identifier; inductive vs transductive). Record the residual score R. S3: Classify X by verification cost: is a candidate answer checkable locally and is the solution unique? If yes, test-time width gains are expected and are credited to the selector, not to the recursion. S4: Separate the three functions of recursion on R: (a) computation: compare with an untied model of equal effective depth and FLOPs; (b) regularisation: compare with an untied model of equal parameters, and vary training-set size; (c) depth at test: run past the training horizon. Credit iteration-as-computation only if (a) favours the tied model with parameters matched, or (c) recovers errors. S5: Report cost on three axes: parameters, training FLOPs (including any training on evaluation tasks), and forward passes per reported answer. Output: the share of A credited to each ingredient, the function of recursion the residual supports, and cost.

with the same checkpoints. If the guide's discrimination and the realised gain rise on the unique-path set, the Maze-Hard failure in Section 6.3 was partly a labelling artefact; if not, it reflects a real limit of the halting head on global verification.

Study 5: a protocol-matched language-model comparison. Give a language model the same 1000 training puzzles, the same augmentations and the same voting, fine-tune it at a FLOP budget matched to the recursive model's training, and report accuracy and cost per answer on Sudoku-Extreme and Maze-Hard. The comparisons in the original papers hold none of these fixed. Two published ingredients suggest the gap would narrow - test-time training lifts language models on ARC by up to six times over fine-tuned baselines (Akyurek et al., 2024), and a recursive scaffold on a pretrained language-model backbone has been reported to outperform earlier recursive models across eight symbolic benchmarks (Manzari et al., 2026) - but no study has run the matched comparison on the puzzle benchmarks.

12. The strongest case against this paper

The best objection is that the decomposition is unfair to the original claim. Nobody said that a 7-million-parameter network solves ARC from its input grids with no help; the TRM paper reports its augmentation, its embeddings and its voting, and the claim was only that this pipeline, with a tiny recursive core, reaches a score that much larger models do not. On that reading, Section 5 attacks a claim nobody made. The reply is that the claim was made in the explanatory sentences that travel with the numbers - that recursion, a latent dynamical system, iterated depth, is what makes tiny networks powerful (Corbett et al., 2026) - and that the identifier ablation and the step-wise readout bear on exactly that sentence. The paper accepts the pipeline result as stated.

A second objection is that the Sudoku evidence of Section 4 is strong and the ARC evidence of Section 5 rests largely on one checkpoint read by one group (Roya-Azar et al., 2025), a checkpoint trained with four latent cycles rather than the six of the paper's best model. That is a fair limit. The identifier result is unlikely to depend on cycle count, since it removes the conditioning the model was trained with; the step-wise result might, and the paper should be read as establishing shallow effective recursion for that checkpoint, with the training-level ablation (Jolicoeur-Martineau, 2025) and the cross-task budget result (Shakhvalieva et al., 2026) as the corroboration. The compute-matched comparison of Gao et al. (2025) runs in the other direction and is the main reason the paper does not conclude that iteration is idle on ARC.

A third objection is that "selection gains" undersells the test-time work. Generating diverse candidates that contain the answer is not trivial, and the anchor's diagnostics predict without labels when its procedure will help, which is a real contribution. The paper agrees. Its claim is about attribution: the measured gain is

bounded by the selector, so the gain is evidence about the selector and the domain's verification cost as much as about the recursion that generated the candidates.

A fourth objection is that the regularisation reading of Section 5.4 is supported mainly by indirect evidence. For ARC that is correct: the one matched-effective-depth puzzle result is on Sudoku-Extreme (Huang et al., 2026), the corroborating study is on code generation (Sirivella et al., 2026), and the rest is an author's suspicion. The paper therefore states regularisation on ARC as an account the evidence fits at least as well as the depth account, not as a finding. Study 1 would decide it.

13. What is not known

It is not known how much of TRM's ARC accuracy survives an inductive protocol without task identifiers and without voting at matched training compute; the closest measurement, 37.50 percent pass@2 under an inductive protocol with identifiers and test-time training, is one run per configuration. It is not known whether weight-tying helps on ARC through computation or through regularisation; no published comparison matches effective depth and parameters at once on that benchmark. It is not known whether the halting head's discrimination predicts test-time gain across tasks, because no paper reports both across more than two benchmarks. It is not known how a language model trained on the same thousand puzzles with the same augmentations would score. And it is not known whether the depth-continuation result on Sudoku extends to instances harder than the training distribution, because the Sudoku-Extreme test set is drawn from the same pool as the training set.

14. Limitations

The analysis covers results published as preprints, most of them from 2026 and few of them peer-reviewed; the conclusions inherit their status. The paper relies on full-text reading of the sources whose numbers it uses and on abstracts for the remainder; where a claim rests on an abstract, the text says what the source reports rather than what it shows. Several key numbers come from single runs (Deliege and Van Droogenbroeck, 2026) or from one released checkpoint (Roya-Azar et al., 2025). The 19-to-39 percent deep-supervision result is reported second-hand, through the TRM paper's citation of an ARC Prize Foundation analysis that is not itself a paper. Benchmarks differ in their maze variants (Maze-Hard against a uniquely solvable set) and in their Sudoku baselines (84.8 to 87.4 percent for nominally the same TRM), so cross-paper gains in Table 1 and Figure 1(b) are each measured against the paper's own baseline and should not be compared to the decimal. The classification column of Table 1 is the author's.

15. Conclusion

A tiny recursive model's puzzle score is a joint product of iterated computation, parameter sharing, a learned selector, a per-task embedding, training on the

evaluation tasks, a thousand-way vote and an optimisation recipe. The published record assigns these shares differently by task. On Sudoku-Extreme, iteration earns direct credit: one-pass models of the same size fail, and running the recursion longer recovers most errors; at matched effective depth, part of that credit is regularisation. On ARC-AGI, the score vanishes without the task identifier, a quarter of it is the vote, with the vote applied the first recursion step already reaches 94 percent of the final score, and the one compute-matched result in favour of weight-tying does not distinguish depth from regularisation. The 2026 test-time gains are real where the model's halting head can check its own answer and absent where it cannot. "Tiny" is true of the parameter count and of nothing else that a comparison with language models would need to hold fixed. The claim that small networks reason through recursion is supported for locally checkable constraint satisfaction; for abstract rule induction it awaits the factorial comparison of Section 11, Study 1.

References

- Akyurek, E., Damani, M., Zweiger, A., Qiu, L., Guo, H., Pari, J., Kim, Y., & Andreas, J. (2024). The Surprising Effectiveness of Test-Time Training for Few-Shot Learning. arXiv:2411.07279. doi:10.48550/arXiv.2411.07279.
- Asadulaev, A., Banerjee, R., Karray, F., & Takac, M. (2025). Latent Reasoning in TRMs is Secretly a Policy Improvement Operator. arXiv:2511.16886. doi:10.48550/arXiv.2511.16886.
- Baek, J., Jo, M., Kim, M., Ren, M., Bengio, Y., & Ahn, S. (2026). Generative Recursive Reasoning. arXiv:2605.19376. doi:10.48550/arXiv.2605.19376.
- Bai, S., Kolter, J. Z., & Koltun, V. (2019). Deep Equilibrium Models. NeurIPS 2019. arXiv:1909.01377. doi:10.48550/arXiv.1909.01377.
- Banino, A., Balaguer, J., & Blundell, C. (2021). PonderNet: Learning to Ponder. arXiv:2107.05407. doi:10.48550/arXiv.2107.05407.
- Bansal, A., Schwarzschild, A., Borgnia, E., Emam, Z., Huang, F., Goldblum, M., & Goldstein, T. (2022). End-to-end Algorithm Synthesis with Recurrent Networks: Logical Extrapolation Without Overthinking. arXiv:2202.05826. doi:10.48550/arXiv.2202.05826.
- Bickov, V. K., & Yorke-Smith, N. (2026). RELAGRAM: Recursive Latent Graph Model for the Capacitated Vehicle Routing Problem. arXiv:2610.15106. doi:10.48550/arXiv.2610.15106.
- Brown, B., Juravsky, J., Ehrlich, R., Clark, R., Le, Q. V., Re, C., & Mirhoseini, A. (2024). Large Language Monkeys: Scaling Inference Compute with Repeated Sampling. arXiv:2407.21787. doi:10.48550/arXiv.2407.21787.
- Chollet, F. (2019). On the Measure of Intelligence. arXiv:1911.01547. doi:10.48550/arXiv.1911.01547.
- Chollet, F., Knoop, M., Kamradt, G., & Landers, B. (2024). ARC Prize 2024: Technical Report. arXiv:2412.04604. doi:10.48550/arXiv.2412.04604.
- Chollet, F., Knoop, M., Kamradt, G., Landers, B., & Pinkard, H. (2025). ARC-AGI-2: A New Challenge for Frontier AI Reasoning Systems. arXiv:2505.11831. doi:10.48550/arXiv.2505.11831.
- Chollet, F., Knoop, M., Kamradt, G., & Landers, B. (2026). ARC Prize 2025: Technical Report. arXiv:2601.10904. doi:10.48550/arXiv.2601.10904.
- Corbett, A., Sood, A., Tzatzopoulou, A., Ramesh, S.-A., & Dodwell, T. (2026). Boosting Inference with Guided Reasoning: Stochastic Exploration for Recursive Models. ICML 2026 Workshop on Structured Probabilistic Inference and Generative Modeling. arXiv:2605.25230. doi:10.48550/arXiv.2605.25230.
- Dehghani, M., Arnab, A., Beyer, L., Vaswani, A., & Tay, Y. (2021). The Efficiency Misnomer. arXiv:2110.12894. doi:10.48550/arXiv.2110.12894.
- Dehghani, M., Gouws, S., Vinyals, O., Uszkoreit, J., & Kaiser, L. (2018). Universal Transformers. ICLR 2019. arXiv:1807.03819. doi:10.48550/arXiv.1807.03819.
- Deliege, A., Beger, C., Van Droogenbroeck, M., & Mitchell, M. (2026). Implicit Rule Induction with Test-Time Task Embeddings in ARC-like Tasks. arXiv:2609.21181. doi:10.48550/arXiv.2609.21181.
- Deliege, A., & Van Droogenbroeck, M. (2026). Pretraining and Test-Time Training Budgets Matter for ARC: A Controlled Study of VARC and TRM. BNAIC/BeNeLearn 2026. arXiv:2610.13660. doi:10.48550/arXiv.2610.13660.
- Dillon, J. V. (2026). Speed is Confidence. arXiv:2601.19085. doi:10.48550/arXiv.2601.19085.
- Fan, Y., Du, Y., Ramchandran, K., & Lee, K. (2024). Looped Transformers for Length Generalization. ICLR 2025. arXiv:2409.15647. doi:10.48550/arXiv.2409.15647.
- Franzen, D., Disselhoff, J., & Hartmann, D. (2025). Product of Experts with LLMs: Boosting Performance on ARC Is a Matter of Perspective. arXiv:2505.07859. doi:10.48550/arXiv.2505.07859.
- Gao, Z., Chen, L., Xiao, Y., Xing, H., Tao, R., Luo, H., Zhou, J., & Dai, B. (2025). Universal Reasoning Model. arXiv:2512.14693. doi:10.48550/arXiv.2512.14693.
- Ge, R., Liao, Q., & Poggio, T. (2025). Hierarchical Reasoning Models: Perspectives and Misconceptions. arXiv:2510.00355. doi:10.48550/arXiv.2510.00355.
- Geiping, J., McLeish, S., Jain, N., Kirchenbauer, J., Singh, S., Bartoldson, B. R., Kailkhura, B., Bhatle, A., & Goldstein, T. (2025). Scaling up Test-Time Compute with Latent Reasoning: A Recurrent Depth Approach. arXiv:2502.05171. doi:10.48550/arXiv.2502.05171.
- Giannou, A., Rajput, S., Sohn, J., Lee, K., Lee, J. D., & Papailiopoulos, D. (2023). Looped Transformers as Programmable Computers. arXiv:2301.13196. doi:10.48550/arXiv.2301.13196.
- Graves, A. (2016). Adaptive Computation Time for Recurrent Neural Networks. arXiv:1603.08983. doi:10.48550/arXiv.1603.08983.
- Hakimi, N. (2026). Form Follows Function: Recursive Stem Model. arXiv:2603.15641. doi:10.48550/arXiv.2603.15641.
- Hu, K., Cy, A., Qiu, L., Ding, X. D., Wang, R., Zhu, Y. E., Andreas, J., & He, K. (2025). ARC Is a Vision Problem! arXiv:2511.14761. doi:10.48550/arXiv.2511.14761.
- Huang, B., Geng, Z., & Kolter, Z. (2026). Equilibrium Reasoners: Learning Attractors Enables Scalable Reasoning. arXiv:2605.21488. doi:10.48550/arXiv.2605.21488.
- Illouz, M. (2026). Resample the Answer: Where to Spend Test-Time Compute in Recursive Reasoning. arXiv:2610.16427. doi:10.48550/arXiv.2610.16427.
- Ingolfsson, T. M., Tahir, W., Tegon, A., Kesting, L., Islamoglu, G., & Benini, L. (2026). Quantizing Recursive Reasoning Models. arXiv:2607.16237. doi:10.48550/arXiv.2607.16237.
- Jolicoeur-Martineau, A. (2025). Less is More: Recursive Reasoning with Tiny Networks. arXiv:2510.04871. doi:10.48550/arXiv.2510.04871.
- Komisarczyk, M., Mathur, S., Kraus, M., Natarajan, S., & Kersting, K. (2026). Recursive Inference Machines for Neural Reasoning. arXiv:2603.05234. doi:10.48550/arXiv.2603.05234.
- Labovich, A. (2026). Stability and Generalization in Looped Transformers. arXiv:2604.15259. doi:10.48550/arXiv.2604.15259.
- Li, W.-D., Hu, K., Larsen, C., Wu, Y., Alford, S., Woo, C., Dunn, S. M., Tang, H., Naim, M., Nguyen, D., Zheng, W.-L., Tavares, Z., Pu, Y., & Ellis, K. (2024). Combining Induction and Transduction for Abstract Reasoning. arXiv:2411.02272. doi:10.48550/arXiv.2411.02272.

- Liao, I., & Gu, A. (2025). ARC-AGI Without Pretraining. arXiv:2512.06104. doi:10.48550/arXiv.2512.06104.
- Mahendrakar, P. (2026a). The Self-Verification Gap: Why Working Hallucination Detectors Are External, and What Real-Time Correction Would Actually Require. Life of Research. doi:10.5281/zenodo.22030500.
- Mahendrakar, P. (2026b). The Crossover Does Not Carry the Claim: Pass@k Inversion, the Reasoning Boundary, and Five Premises the RLVR Debate Leaves Unstated. Life of Research. doi:10.5281/zenodo.22063128.
- Mahendrakar, P. (2026c). What Counts as a Control? A Scaffold Effect Is a Contrast Conditional on What Was Held Fixed, the 2026 Controlled Studies Each Held Something Different, and Only a Declared Spend Interval Makes a Harness Claim Refutable. Life of Research. doi:10.5281/zenodo.23306459.
- Manzari, O. N., Lajoie, G., & Rivaz, H. (2026). Recursive Vision Language Models for General Symbolic Reasoning. arXiv:2608.01534. doi:10.48550/arXiv.2608.01534.
- McGovern, R. K. (2025). Test-time Adaptation of Tiny Recursive Models. arXiv:2511.02886. doi:10.48550/arXiv.2511.02886.
- Palm, R. B., Paquet, U., & Winther, O. (2017). Recurrent Relational Networks. NeurIPS 2018. arXiv:1711.08028. doi:10.48550/arXiv.1711.08028.
- Ren, Z., & Liu, Z. (2026). Are Your Reasoning Models Reasoning or Guessing? A Mechanistic Analysis of Hierarchical Reasoning Models. arXiv:2601.10679. doi:10.48550/arXiv.2601.10679.
- Rodrigues, L. R., & Kang, J. (2026). Dissecting Hierarchical Reasoning Models: A Mechanistic Study. arXiv:2609.22197. doi:10.48550/arXiv.2609.22197.
- Roye-Azar, A., Vargas-Naranjo, S., Ghai, D., Balamurugan, N., & Amir, R. (2025). Tiny Recursive Models on ARC-AGI-1: Inductive Biases, Identity Conditioning, and Test-Time Compute. arXiv:2512.11847. doi:10.48550/arXiv.2512.11847.
- Saunshi, N., Dikkala, N., Li, Z., Kumar, S., & Reddi, S. J. (2025). Reasoning with Latent Thoughts: On the Power of Looped Transformers. ICLR 2025. arXiv:2502.17416. doi:10.48550/arXiv.2502.17416.
- Schwarzschild, A., Borgnia, E., Gupta, A., Huang, F., Vishkin, U., Goldblum, M., & Goldstein, T. (2021). Can You Learn an Algorithm? Generalizing from Easy to Hard Problems with Recurrent Networks. arXiv:2106.04537. doi:10.48550/arXiv.2106.04537.
- Sghaier, A., Parviz, A., & Jolicoeur-Martineau, A. (2026). Probabilistic Tiny Recursive Model. arXiv:2605.19943. doi:10.48550/arXiv.2605.19943.
- Shakhvalieva, Y., Kharchev, D., Bezrukov, V., Fedorova, I., Bocharov, D., Oseledets, I., & Ternovskii, V. (2026). What Limits Recursive Reasoning Models: Optimization, Architecture and Test-Time Scaling. arXiv:2609.39967. doi:10.48550/arXiv.2609.39967.
- Sirivella, A., Newaz, A., & Melo, G. (2026). Evaluating Tiny Recursive Models Across Training for Code Generation. arXiv:2608.29376. doi:10.48550/arXiv.2608.29376.
- Stroebel, B., Kapoor, S., & Narayanan, A. (2024). The Limits of Inference Scaling Through Resampling. arXiv:2411.17501. doi:10.48550/arXiv.2411.17501.
- Wang, G., Li, J., Sun, Y., Chen, X., Liu, C., Wu, Y., Lu, M., Song, S., & Abbasi Yadkori, Y. (2025). Hierarchical Reasoning Model. arXiv:2506.21734. doi:10.48550/arXiv.2506.21734.
- Wang, P.-W., Donti, P. L., Wilder, B., & Kolter, Z. (2019). SATNet: Bridging deep learning and logical reasoning using a differentiable satisfiability solver. ICML 2019. arXiv:1905.12149. doi:10.48550/arXiv.1905.12149.
- Wang, W., & Reid, F. (2026). Tiny Recursive Reasoning with Mamba-2 Attention Hybrid. arXiv:2602.12078. doi:10.48550/arXiv.2602.12078.
- Wang, X., Wei, J., Schuurmans, D., Le, Q., Chi, E., Narang, S., Chowdhery, A., & Zhou, D. (2022). Self-Consistency Improves Chain of Thought Reasoning in Language Models. ICLR 2023. arXiv:2203.11171. doi:10.48550/arXiv.2203.11171.
- Waugh, J. (2026). Pencil Puzzle Bench: A Benchmark for Multi-Step Verifiable Reasoning. arXiv:2603.02119. doi:10.48550/arXiv.2603.02119.