TINY RECURSIVE MODELS UNDERUSE THEIR DEPTH

Anonymous authors
Paper under double-blind review

ABSTRACT

Tiny recursive models (TRMs) solve hard reasoning benchmarks by repeatedly applying a small weight-tied network to a pair of carried states, and improvements to this family have so far come from iterating more: extra recursion cycles, extra supervision steps, more effective depth. However, a failure case emerges: the recursion reaches a fixed point quickly. Successive iterations change the carried states less and less, so much of the depth budget is spent after the recursion has effectively reached its fixed point, and the added depth is ineffective. We show that a larger residual stream alleviates this failure case. Adopting Virtual Width Networks (VWN), we widen the state the recursion carries several-fold while leaving backbone compute unchanged, so each iteration has room to write something the previous one did not and the fixed point becomes harder to reach. Widening also makes the depth TRMs already run useful: held-out sweeps show the wide model's margin over the narrow one growing with the number of supervision steps allowed at evaluation when many candidate answers are scored, while the narrow model's candidate pool stops improving early. We validate on ARC-AGI-1, ARC-AGI-2, and Sudoku-Extreme with 7M-parameter models under matched configurations. The resulting VWN+TRM improves ARC-AGI-1 pass@2 by at least 7.1 points over both TRM baselines we trained (the upstream and the depth-matched configuration), improves ARC-AGI-2 pass@2 by 3.8 points over the matched baseline (1.2 points, or 16% relative, over the published TRM result), and adds 11.2 points of exact accuracy on Sudoku-Extreme.

1 INTRODUCTION

Tiny recursive models solve reasoning benchmarks that defeat much larger networks. The Hierarchical Reasoning Model (HRM; Wang et al., 2025) and the Tiny Recursive Model (TRM; Jolicoeur-Martineau, 2025) reach strong results on ARC-AGI (Chollet, 2019; Chollet et al., 2025) and Sudoku-Extreme with networks of a few million parameters by applying one weight-tied core to a pair of carried states over many cycles, under deep supervision with learned halting (Graves, 2016). Later work follows the same direction: URM (Gao et al., 2025) credits the gains to the recurrent inductive bias, Equilibrium Reasoners (Huang et al., 2026) unroll to the equivalent of tens of thousands of layers at test time, and looped vision transformers scale iteration counts on visual ARC (Shu et al., 2026). In each case the improvement comes from running the recursion longer. Running it longer, however, exposes a failure case. The recursion reaches a fixed point quickly (Figure 1): successive iterations change the carried states less and less, and much of the depth budget is spent after the recursion has effectively reached its fixed point.

We show that a larger residual stream alleviates this failure case. A weight-tied update keeps doing useful work only while it can keep moving its state, and that state is the model's entire memory between iterations: every intermediate result must be written into a carried state of some fixed width D per token, which can hold only about D independent quantities per token (Dambre et al., 2012). A narrow residual stream therefore fills within a few iterations; once new writes can only interfere with content the answer still needs, the update that best preserves the answer is to change almost nothing, and the trajectory settles onto an attractor (Huang et al., 2026). The recursion therefore stops when the residual stream is full, whether or not the task is finished. Widening the residual stream raises this budget linearly, so useful change persists across more supervision steps. To widen at essentially iso-FLOP cost we adopt Virtual Width Networks (VWN; Seed et al., 2025), adapting them from feed-forward language-model pretraining to the weight-tied recursive setting: both carried states live at several times the backbone width, and generalized hyper-connections (Zhu et al.,

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.

2025) compress to the backbone width for every attention and feed-forward sublayer and expand back, leaving backbone compute, attention shape, and sequence length untouched (Section 4). This account predicts that the wider model's margin should grow with the step budget, and our held-out sweeps confirm this at large candidate budgets, that is, at large k in pass@k (Section 6.3).

The wider model is more accurate on every benchmark at every candidate budget we measured. On ARC-AGI-1, VWN+TRM reaches 52.38% pass@2 and 46.75% pass@1, above both TRM baselines we trained: the depth-matched H6/L9 TRM (45.25% and 37.75%) and the upstream H3/L4 TRM, the stronger of the two at small k (44.63% and 39.50%), so the smaller of the two gains is 7.1 pass@2 and 7.3 pass@1 points. It improves ARC-AGI-2 from 5.28% to 9.03% pass@2 (4.44% to 8.61% pass@1; 1.2 points, or 16% relative, over the published TRM result, Section 5.1), and lifts a matched TRM on Sudoku-Extreme from 69.5% to 80.7% exact accuracy, at a cost of 5.5% additional parameters and no added layers or cycles. The wider model also halts in fewer supervision steps. Interestingly, its margin *increases* with the number of supervision steps: in held-out sweeps of the step budget, the pass@100 margin grows monotonically from $+6.9$ to $+8.8$ points across budgets 1 to 16, because the narrow model's candidate pool saturates by four steps while the wide model's is still improving at sixteen (Sections 5.2 and 6.3). Width and iteration are therefore complements: a wider residual stream keeps additional supervision steps useful, and the value of widening increases with their number.

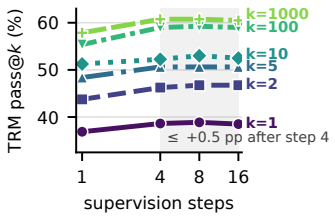

Figure 1: **A TRM stops improving after a few supervision steps.** ARC-AGI-1 pass@k (gold grid among the top k ranked candidates) of the depth-matched H6/L9 TRM at step budgets of 1, 4, 8, and 16: flat beyond 4 (at most $+0.5$ points) though training exposes 16.

2 Related work

Tiny recursive reasoners. HRM (Wang et al., 2025) introduced a two-timescale recurrent architecture with deep supervision and learned halting that solves ARC-AGI, Sudoku-Extreme, and Maze tasks with a 27M-parameter network. TRM (Jolicoeur-Martineau, 2025) simplified HRM to a single weight-tied two-layer block over an answer state and a latent state, halving parameters and improving accuracy; we build directly on TRM and adopt its protocol. URM (Gao et al., 2025) reframes this family as universal transformers (Dehghani et al., 2019), attributing the gains to recurrent inductive bias plus strong nonlinearities. Equilibrium Reasoners (Huang et al., 2026) interpret TRM-style models as learned attractor dynamics and scale test-time iterations and restarts, reaching over 99% on Sudoku-Extreme; LoopViT (Shu et al., 2026) applies looped computation to visual ARC. These works all scale the amount of iteration. We instead vary the width of the carried states.

Depth recurrence and looped transformers. Weight-tied depth has long been studied as a compute-for-parameters trade (Dehghani et al., 2019; Graves, 2016), with deep equilibrium models taking the infinite-depth limit (Bai et al., 2019; Anil et al., 2022; Georgiev et al., 2024). Looped transformers are provably expressive (Giannou et al., 2023; Saunshi et al., 2025), generalize from easy to hard instances (Schwarzschild et al., 2021), and benefit language models at scale (Geiping et al., 2025; Bae et al., 2025); chain-of-thought itself is recurrence through the token stream (Merrill & Sabharwal, 2024). Our results do not contradict the value of recurrence (Section 6.4); they show that depth beyond the TRM operating point does not help at this scale, whereas width does.

Widening the residual stream. Hyper-connections (Zhu et al., 2025) replace the residual stream's additive updates (Elhage et al., 2021) with learned mixing over multiple residual lanes. VWN (Seed et al., 2025) generalizes this into an explicit width-scaling axis: embeddings at virtual width, per-sublayer generalized hyper-connections (GHC) that compress to the backbone width and expand back, and a reduction layer at the output, yielding log-linear loss improvements in language-model pretraining at nearly constant backbone compute. We port this machinery, including dynamic GHC, into a weight-tied recursive setting where the same GHC modules are reused across dozens of cycles and sixteen supervision steps, a regime VWN was not designed for, and find its benefits transfer intact. Sapunov (2026) is the closest prior work: a single-block universal transformer on Sudoku-

Extreme is unusable without memory tokens that enlarge the persistent state, and memory trades off against pondering depth. We test the same trade directly at the residual-stream level, and on ARC.

3 BACKGROUND: TINY RECURSIVE MODELS

Notation. Vectors and matrices are bold ($\boldsymbol{z}$, $\boldsymbol{B}$) and scalars italic. A TRM processes a sequence of S tokens and carries two states across its recursion: an answer state $\boldsymbol{z}_H$, decoded into the prediction, and a latent state $\boldsymbol{z}_L$ acting as a scratchpad, each an $S \times D$ matrix, where D is the width of the backbone network. One weight-tied network f_θ is applied for H outer cycles, each containing L inner cycles; we write HH/LL for such a configuration (H6/L9 has $H = 6$ and $L = 9$), and H_{grad} is the number of outer cycles through which gradients flow. A training sequence receives up to N_{sup} supervision steps, and q is the logit of the halting head. Our widened model carries its states at a virtual width $D' > D$ (Section 4); the maps that change width are the input lift $\boldsymbol{E}$ ($D \to D'$), the compression $\boldsymbol{B}$ ($D' \to D$), the expansion $\boldsymbol{U}$ ($D \to D'$), the lane-mixing residual $\boldsymbol{M}$ ($D' \to D'$), and the output reduction $\boldsymbol{R}$ ($D' \to D$).

Architecture and recursion. In a TRM (Jolicoeur-Martineau, 2025) the network f_θ — here two post-norm (Xiong et al., 2020) transformer blocks (Vaswani et al., 2017) with self-attention and SwiGLU (Shazeer, 2020) at width $D = 512$ — performs one *supervision step* as a nested recursion. Let $\boldsymbol{z}_H^{(h)}$ denote the answer state after outer cycle h and $\boldsymbol{z}_L^{(h,t)}$ the latent state after t inner cycles of outer cycle h, with $\boldsymbol{z}_L^{(h,0)} = \boldsymbol{z}_L^{(h-1,L)}$. For $h = 1, \dots, H$, the latent state is refined L times and the answer state once:

$$\boldsymbol{z}_L^{(h,t)} = f_\theta\big(\boldsymbol{z}_L^{(h,t-1)},\, \boldsymbol{z}_H^{(h-1)} + \boldsymbol{x}\big), \quad t = 1, \dots, L, \qquad \boldsymbol{z}_H^{(h)} = f_\theta\big(\boldsymbol{z}_H^{(h-1)},\, \boldsymbol{z}_L^{(h,L)}\big), \tag{1}$$

where $\boldsymbol{x}$ is the embedded input and the second argument of f_θ is added into the residual stream as an input injection; the step's outputs $\boldsymbol{z}_H^{(H)}$ and $\boldsymbol{z}_L^{(H,L)}$ are carried into the next step. One supervision step therefore makes $H(L+1)$ calls to f_θ, that is, $2H(L+1)$ transformer blocks, which we call its *effective depth*. Following TRM, gradients flow only through the last H_{grad} outer cycles; the first $H - H_{\text{grad}}$ run without gradient tracking (TRM uses $H_{\text{grad}} = 1$; our deep configurations use 2).

Deep supervision and halting. The recursion (1) is wrapped in up to $N_{\text{sup}} = 16$ supervision steps: after each step the answer head is trained with stablemax cross-entropy (Prieto et al., 2025), the states are detached and carried forward, and a halting head with logit $q = q(\boldsymbol{z}_H)$ is trained to predict whether the current decoded answer is exactly correct (Graves, 2016; Wang et al., 2025). During training a sequence halts once $q > 0$, with an exploration scheme forcing occasional longer rollouts; during evaluation halting is disabled and all 16 steps always run, so evaluation compute is fixed; we call the number of steps an evaluation is allowed the *step budget*. Optimization uses Adam-atan2 (Everett et al., 2024; Kingma & Ba, 2015) with an exponential moving average (EMA) of the weights, and training data are heavily augmented (dihedral and color transforms for ARC; band and digit permutations for Sudoku-Extreme).

Evaluation protocol. We report exact accuracy for Sudoku-Extreme, that is, exact match of the whole solution sequence. For ARC we follow the repository's evaluator, whose semantics we state precisely because they differ from sampling-based pass@k (Chen et al., 2021): every augmented prediction is inverse-transformed to the original grid frame; candidate grids are hash-deduplicated and ranked by vote count and then by mean halting logit q; and pass@k asks whether the gold grid appears among the top k *unique ranked candidates*; we call k the *candidate budget*. The score is averaged within tasks and then macro-averaged over the 400 (ARC-AGI-1) or 120 (ARC-AGI-2) tasks. The evaluator additionally accumulates candidates *across* the ten training-time evaluations, so the ith reported value is a cumulative vote ensemble over checkpoints $1, \dots, i$; all "final" ARC numbers below are cumulative ensembles over all ten evaluation checkpoints, applied identically to baselines and VWN models (independently re-derived top-2 receipts in Appendix A).

4 VWN+TRM: WIDENING THE CARRIED STATES

Our hypothesis is that TRM performance is limited by the capacity of the carried states rather than by how many times they are transformed. The direct test, raising D, is quadratic in cost, since the parameters and FLOPs of attention and of the feed-forward network grow as D^2. VWN (Seed et al.,

```
x = E(embed(tokens))                        # lift the input from width D to D'

y, z = y_init, z_init                       # carried states z_H, z_L at width D'
for step in range(N_sup):                   # deep supervision (as in TRM)
    for cycle in range(H):                  # outer cycles; grads on last H_grad
        for i in range(L):                  # inner latent recursion
            z = net(x, y, z)
        y = net(y, z)                       # refine the answer

    y_out = R(group_norm(y))                # back to width D for the heads

    y_hat, q = output_head(y_out), Q_head(y_out)
    train on (y_hat, q); break if q > 0     # TRM loss and halting

def net(*inputs):                           # weight-tied core, states at D'
    h = sum(inputs)
    for g in (attn, ffn, attn, ffn):        # each sublayer g runs at width D
        h = rms_norm(M(h) + U(g(B(h))))     # hyper-connection wrap
    return h
```

Figure 2: **VWN+TRM in pseudocode.** Condensed from the TRM training loop (Jolicoeur-Martineau, 2025, Figure 3), whose schedule is unchanged: H outer cycles of L inner latent updates per supervision step, up to N_{sup} steps, halting once the halting logit q is positive (TRM's `y`, `z` are our $\boldsymbol{z}_H$, $\boldsymbol{z}_L$). Red boxes mark the three VWN changes, as in Figure 3: the input lift $\boldsymbol{E}$ ($D \to D'$), the hyper-connection wrap $\boldsymbol{M}\boldsymbol{h}' + \boldsymbol{U}\, g(\boldsymbol{B}\boldsymbol{h}')$ around every sublayer g (the GHC update (2)), and the output reduction $\boldsymbol{R}$ ($D' \to D$) before the heads.

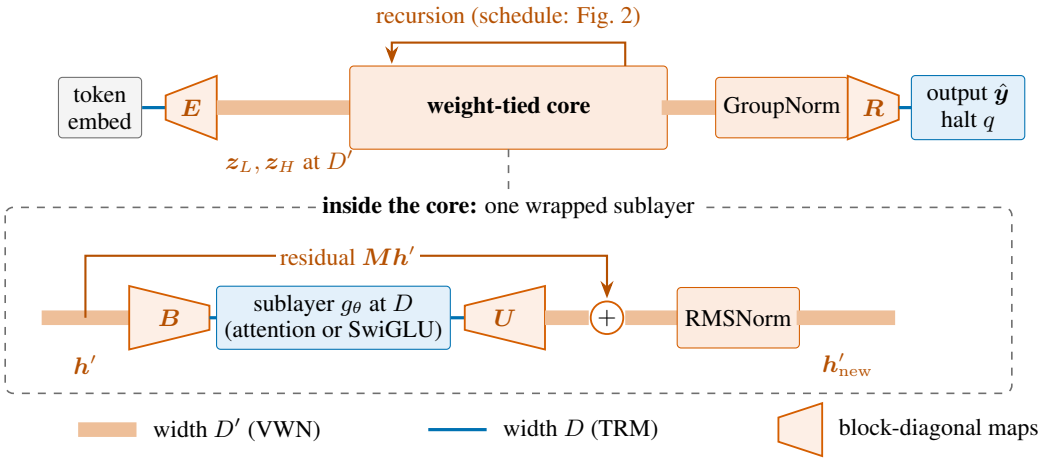

Figure 3: **VWN+TRM.** *Top:* one supervision step. The input lift $\boldsymbol{E}$ takes token embeddings from the backbone width D to the virtual width $D' = 5D$; the carried states $\boldsymbol{z}_L$, $\boldsymbol{z}_H$ stay at D' through the recursion (schedule in Figure 2); grouped normalization and the output reduction $\boldsymbol{R}$ return the answer state to D for the output head and the halting head (logit q). *Bottom:* each sublayer g_θ of the weight-tied core is wrapped by a dynamic generalized hyper-connection, the GHC update (2): $\boldsymbol{B}$ compresses the input $\boldsymbol{h}'$ to width D, g_θ runs at D, $\boldsymbol{U}$ expands its output to D', the lane-mixing residual $\boldsymbol{M}\boldsymbol{h}'$ is added, and RMSNorm follows at D'. Orange bands carry width D', blue lines width D; trapezoids are block-diagonal maps.

2025) offers the same increase in state capacity at nearly constant backbone cost, and we adapt it to the weight-tied, twice-recurrent TRM setting as follows (Figure 3; pseudocode in Figure 2).

Virtual-width carries. Split the backbone width into m segments of size $d = D/m$, and let the *virtual width* be $D' = nd$ for some $n \geq m$; a virtual-width vector is thus n segments $\boldsymbol{h}' = (\boldsymbol{h}'_1, \ldots, \boldsymbol{h}'_n)$ with each $\boldsymbol{h}'_j \in \mathbb{R}^d$; VWN calls these segments *lanes*. Both carried states now live at virtual width, $\boldsymbol{z}_H, \boldsymbol{z}_L \in \mathbb{R}^{S \times D'}$, as do their learned initializations, and they persist at D' across all cycles *and* supervision steps. We use $m = 8$ and $n = 40$, so $d = 64$ and $D' = 2560 = 5D$.

Generalized hyper-connections around every sublayer. Each attention and each feed-forward sublayer of the weight-tied core is wrapped in its own *generalized hyper-connection* (GHC), which has three learned matrices acting on segments: a compression $\boldsymbol{B} \in \mathbb{R}^{m\times n}$, an expansion $\boldsymbol{U} \in \mathbb{R}^{n\times m}$, and a lane-mixing residual $\boldsymbol{M} \in \mathbb{R}^{n\times n}$. The compression forms m segments $\boldsymbol{h}_i = \sum_{j=1}^{n} B_{ij}\, \boldsymbol{h}'_j$ and concatenates them into a backbone-width vector $\boldsymbol{h} \in \mathbb{R}^D$; the sublayer g_θ (attention or SwiGLU) runs entirely at width D; and its output, re-segmented into m pieces, is expanded back and added to the mixed residual:

$$\boldsymbol{h}'_{\text{new}} = \boldsymbol{M}\, \boldsymbol{h}' + \boldsymbol{U}\, g_\theta(\boldsymbol{h}). \tag{2}$$

The usual width-D residual inside the core is removed, and RMSNorm (Zhang & Sennrich, 2019) is applied to $\boldsymbol{h}'_{\text{new}}$ at width D'. At initialization $\boldsymbol{M}$ is the identity, $\boldsymbol{B}$ is a cyclic gather ($B_{ij} = [\, j \bmod m = i\,]$ in Iverson-bracket notation), and $\boldsymbol{U} = [\, \boldsymbol{I}_m;\, \boldsymbol{0}\,]$, so the network initially computes the same function as the original TRM with the extra lanes idle. We use *dynamic* GHC (Seed et al., 2025): for every token, each of the three matrices receives a low-rank, input-conditioned correction (a learned scale times the $\tanh$ of a linear map of the normalized input). Because f_θ is weight-tied, the *same* four GHC modules are reused at every one of the $H(L+1)$ calls of every supervision step; the dynamic corrections let lane routing differ across those reuses even though the static matrices are shared.

Width at the boundaries. Token embeddings are formed at D and lifted to D' by the input lift $\boldsymbol{E}$, a block-diagonal expansion with block size 32, as is the puzzle embedding after reshaping to its 16 prefix positions, so the input injection $\boldsymbol{x}$ in the recursion (1) is also at D'. At the output, grouped normalization (D'/D groups; Wu & He, 2018) followed by the output reduction $\boldsymbol{R}$ (block-diagonal) to width D feeds the answer head; an independent normalize-and-reduce feeds the halting head; and rotary position encodings act inside attention at width D, unchanged.

Cost. Backbone parameters and FLOPs are untouched, and attention operates at width D at unchanged sequence length, so there is no quadratic penalty. Each GHC module has $2mn + n^2 + n(m+n) + d(2m+n) = 8{,}064$ parameters, and the total number of added parameters is

$$\underbrace{2 \cdot 32D'}_{\text{token and puzzle lifts}} \;+\; \underbrace{2\,(32D' + 2D')}_{\text{output reductions and norms}} \;+\; \underbrace{4 \cdot 8{,}064 + 2D'}_{\text{four GHC modules and norms}} \;=\; 375{,}296, \tag{3}$$

which is 5.50% of the $6{,}829{,}058$-parameter baseline (breakdown in Appendix B). The runtime overhead is one small batched contraction per sublayer; we make no wall-clock claim because our baseline and VWN runs used different GPU allocations (Section 5.1).

5 Experiments

5.1 Setup

Tasks and data. We evaluate on (i) **ARC-AGI-1** (Chollet, 2019): training + evaluation-concept splits with $1000\times$ augmentation following Wang et al. (2025) (400 scored tasks; 385,815 augmented evaluation rows); (ii) **ARC-AGI-2** (Chollet et al., 2025): the analogous `training2/evaluation2` setup (120 tasks; 118,766 rows); and (iii) **Sudoku-Extreme** (Wang et al., 2025): 1,000 training puzzles augmented $1000\times$, 422,786 held-out puzzles, exact-match scoring.

Matched pairs. For ARC-AGI-1 we report two TRM baselines: the depth-matched H6/L9 configuration and the upstream H3/L4 configuration, which is the stronger TRM at small k (Section 6.1); every ARC-AGI-1 gain below is stated against both. Each matched comparison trains a TRM and a VWN+TRM from identical configurations, differing *only* in the VWN modules ($m = 8$ backbone segments, $n = 40$ virtual lanes, dynamic GHC; Section 4). ARC uses the deep configuration H6/L9 with $H_{\text{grad}} = 2$ (effective depth 120 per step); Sudoku-Extreme uses the upstream H3/L6 with $H_{\text{grad}} = 1$ (effective depth 42). All runs: backbone width $D = 512$, 2 layers, 8 heads, expansion 4, RoPE (Su et al., 2024), 16-position puzzle embedding, $N_{\text{sup}} = 16$, seed 0, global batch 768, identical optimization (Adam-atan2 (Everett et al., 2024), LR 10^{-4}, EMA 0.999 (Polyak & Juditsky, 1992)). ARC training runs $\sim$518k (ARC-AGI-1) and $\sim$724k (ARC-AGI-2) training steps, Sudoku 65,100, each with 10 evaluations; full configurations in Appendix F.

Table 1: **Final ARC results** (seed 0; %). VWN+TRM and the depth-matched TRM use H6/L9 (6 outer, 9 inner recursion cycles per step); for ARC-AGI-1 the upstream H3/L4 TRM, the stronger TRM at small k, is also shown (final evaluation; its trajectory was not logged), with gains against both. Pass@k: gold grid among the top k unique ranked candidates, accumulated over the ten evaluation checkpoints, the cumulative ensemble of Section 3. VWN+TRM adds $+5.5\%$ parameters and no depth, yet improves every pass@k on both benchmarks.

				Cumulative pass@k (%) ↑					
Dataset	Model	Config	Params	k=1	2	5	10	100	1000
ARC-AGI-1	TRM	H3/L4	6.83M	39.50	44.63	48.50	51.63	59.00	64.50
	TRM	H6/L9	6.83M	37.75	45.25	51.38	54.13	63.38	67.75
	VWN+TRM	H6/L9	7.20M	**46.75**	**52.38**	**56.75**	**59.50**	**71.00**	**74.38**
	Δ *vs. H3/L4 (pp)*			+7.25	+7.75	+8.25	+7.88	+12.00	+9.88
	Δ *vs. H6/L9 (pp)*		*+5.5%*	+9.00	+7.13	+5.38	+5.38	+7.63	+6.63
ARC-AGI-2	TRM*	H6/L9	6.83M	4.44	5.28	8.75	10.42	16.25	18.75
	VWN+TRM	H6/L9	7.20M	**8.61**	**9.03**	**13.06**	**15.00**	**18.89**	**22.50**
	Δ *vs. H6/L9 (pp)*		*+5.5%*	+4.17	+3.75	+4.31	+4.58	+2.64	+3.75

* Does not reproduce the 7.8% two-attempt ARC-AGI-2 accuracy reported by Jolicoeur-Martineau (2025); see the scope note in Section 5.1.

Scope of the claims. Every number is single-seed (seed 0); per-checkpoint trajectories (Figure 4, Appendix C) stand in for seed error bars. Baseline and VWN runs received different GPU allocations (16 versus 32 for ARC and 1 versus 8 for Sudoku-Extreme) with identical global batch and step counts, so we make *no* wall-clock or throughput claims. ARC pass@k is the cumulative cross-checkpoint vote ensemble defined in Section 3, applied identically to both models. Under this deep configuration and protocol our TRM baseline reaches 5.3 pass@2 on ARC-AGI-2, and we were unable to reproduce the 7.8% two-attempt accuracy reported by Jolicoeur-Martineau (2025) in our setup; measured against that published number rather than our own baseline, VWN+TRM's 9.03 pass@2 is a gain of 1.2 points (16% relative); the same pipeline at the upstream H3/L4 configuration does match their reported ARC-AGI-1 accuracy (44.62 against their reported 44.6 pass@2, Section 6.1), and every claim is stated against the depth-matched pair (ARC-AGI-1 gains also against the upstream TRM).

5.2 Main results: ARC-AGI

Table 1 gives the headline comparison. On ARC-AGI-1, VWN+TRM reaches 52.38 pass@2 and 46.75 pass@1: 7.1 and 9.0 points above the depth-matched H6/L9 TRM, and 7.8 and 7.3 points above the upstream H3/L4 TRM, which is the stronger baseline at $k \leq 2$ but the weaker one at $k \geq 5$ (Table 1); on the much harder ARC-AGI-2 it improves pass@2 from 5.28 to 9.03 and pass@1 from 4.44 to 8.61. The improvement holds at every candidate budget (Figure 4c,d), by at least 5.4 and at most 12.0 points from pass@1 to pass@1000 on ARC-AGI-1 against either TRM baseline, so the wider model produces better candidate pools rather than only a better ranking of the same pool. It also holds throughout training (Figure 4a,b): VWN+TRM leads at all ten evaluation checkpoints on ARC-AGI-1 and at nine of ten on ARC-AGI-2, and its margin grows over training (+3.25 pass@1 over the depth-matched TRM at the first ARC-AGI-1 checkpoint, +9.0 at the last), as VWN's gains also grow with training in language-model pretraining (Seed et al., 2025). Underlying per-example metrics move the same way (Appendix E).

5.3 Main results: Sudoku-Extreme

Table 2 and Figure 6 show the matched comparison at H3/L6: VWN+TRM reaches 80.70% exact accuracy against the TRM's 69.52%, with the gap already large at 19.5k training steps (60.2 against 35.7); the wide model reaches the baseline's final accuracy in roughly half the training steps.

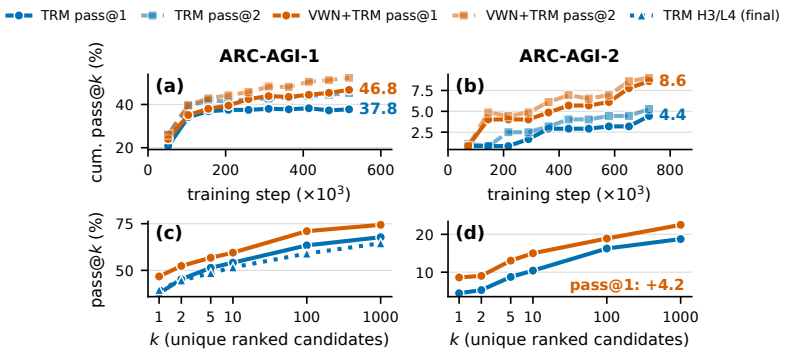

Figure 4: **ARC-AGI results overview. (a, b)** Cumulative pass@1/2 over training for VWN+TRM and the depth-matched TRM; each point ensembles all checkpoints up to that step, and the gap widens with training on both benchmarks. In **(c)** the dotted line is the upstream H3/L4 TRM's final evaluation. **(c, d)** Final cumulative pass@k against k, the number of unique ranked candidates (log axis): width helps at every candidate budget, so the candidate pools themselves are better.

Table 2: **Sudoku-Extreme at training step 65,100** (seed 0; exact accuracy, %). HH/LL denotes H outer and L inner recursion cycles per supervision step. VWN+TRM beats the matched TRM by 11.2 points.

Model	Depth	Params	GPUs	Exact (%) ↑	Token (%) ↑
TRM	H3/L6	6.83M	1	69.52	89.03
VWN+TRM	H3/L6	7.20M	8	**80.70**	**92.65**

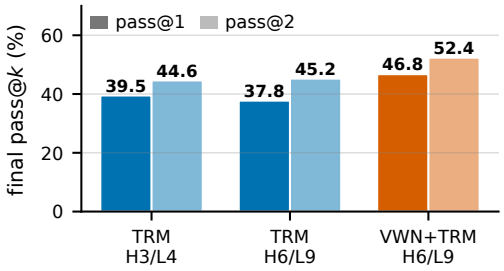

Figure 5: **Depth vs. width at 7M parameters on ARC-AGI-1.** Quadrupling the effective depth per supervision step, from H3/L4 to H6/L9 (H outer and L inner recursion cycles; $2H(L{+}1) = 30$ versus 120 transformer blocks), does not improve the plain TRM, while width at fixed depth adds 9.0 and 7.1 points (7.3 and 7.8 over the shallower H3/L4 TRM).

6 Analysis

This section tests the claim that the width of the carried state, rather than the amount of iteration, is the binding constraint. Section 6.1 shows that added recursion depth does not help; Section 6.2 that the supervision steps are barely used; Section 6.3 that only the wide model benefits from later iterations; and Section 6.4 that a pseudo-token realization of extra capacity underperforms the GHC wiring, while removing recursion collapses the gain.

6.1 Depth scaling saturates on ARC-AGI-1

Figure 5 isolates the two scaling axes on ARC-AGI-1. The completed upstream-style TRM at H3/L4 (effective depth 30 per supervision step) finishes at 39.50 pass@1 and 44.62 pass@2, while the four times deeper H6/L9 TRM finishes at 37.75 and 45.25 under the same data, batch, and

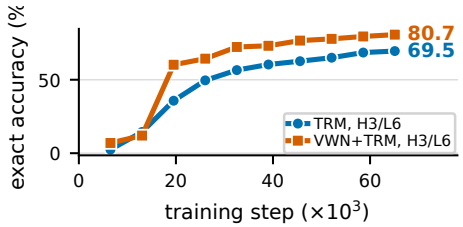

Figure 6: **Sudoku-Extreme over training** (exact accuracy, %). At matched H3/L6 depth (3 outer and 6 inner cycles), VWN+TRM (orange) leads the TRM (blue) throughout training and finishes 11.2 points higher.

Dataset	Model	Steps used (of 16)	halting-head acc. (%) train	eval
ARC-1	TRM	1.31	98.5	74.3
ARC-1	VWN+TRM	**1.19**	99.4	63.1
ARC-2	TRM	1.54	95.8	87.8
ARC-2	VWN+TRM	**1.25**	97.6	82.5
Sudoku	TRM	8.46	100.0	100.0
Sudoku	VWN+TRM	**5.52**	100.0	100.0

Table 3: **Halting statistics.** *Steps used*: average supervision steps taken before halting at the end of training (of $N_{\text{sup}} = 16$; evaluation always runs all 16). *halting-head accuracy*: accuracy of the halting head's exact-correctness prediction on training data and on held-out augmented examples.

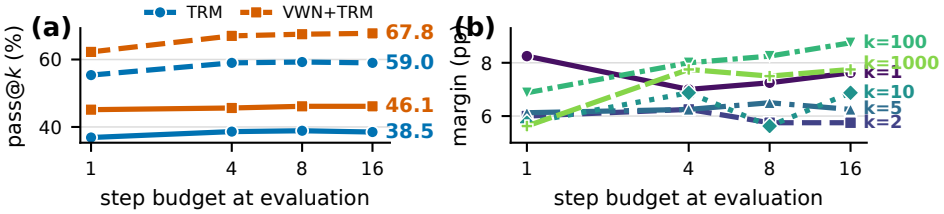

Figure 7: **Held-out step-budget sweep on ARC-AGI-1.** VWN+TRM and the depth-matched H6/L9 TRM at their final EMA checkpoints, evaluated with a budget of 1, 4, 8, or 16 supervision steps (single checkpoint, no cross-checkpoint voting, so absolute levels differ from Table 1). **(a)** Pass@k of TRM (blue) and VWN+TRM (orange) at $k = 1$ (solid) and $k = 100$ (dashed): the wide model's first step alone beats the narrow model's best, and only the wide model keeps rising past 4 steps. **(b)** The *margin*, VWN+TRM pass@k minus TRM pass@k in percentage points, for each k: positive in every cell ($+5.6$ to $+8.8$), steepening with k across budgets and monotone at pass@100.

step budget: pass@1 is slightly *lower*, pass@2 is within noise, and only the large-k tail benefits (pass@1000 rises from 64.5 to 67.8). Quadrupling depth therefore does not improve pass@1 or pass@2, whereas width adds 9.0 pass@1 points at fixed depth and 7.3 over the shallower H3/L4 TRM. Added depth at this scale does not help, while added width helps on every benchmark we ran, consistent with iterations that have nothing new to carry.

6.2 Deep supervision collapses onto the first steps

TRM's second depth axis is the outer deep-supervision loop, and the training statistics show it is barely used at convergence (Table 3). Although 16 supervision steps are available and evaluation always runs all of them, the learned halting head ends training taking on average only 1.31 (TRM) and 1.19 (VWN+TRM) steps per ARC-AGI-1 training sequence, 1.54 and 1.25 on ARC-AGI-2, and the halting head's decision after the first step already predicts exact correctness with 95.8–99.4% accuracy. By the end of training the first supervision step produces what is essentially its final answer for almost every training sequence, so the gains of deep supervision come from the earliest steps. Sudoku-Extreme, whose held-out puzzles are far harder than ARC's memorized training distribution, retains more stepwise refinement (8.46 steps for TRM) , but the wide model uses fewer steps (5.52). In every pairing the wider model reaches the same or better accuracy in fewer supervision steps, consistent with the added capacity being used in the first supervision step, as a bandwidth-bottleneck account predicts (Sapunov, 2026).

Not every metric favors the wider model. On ARC, VWN+TRM's evaluation-time halting-head accuracy is lower than the baseline's (63.1% against 74.3%; Table 3) even as its task accuracy is higher; on Sudoku-Extreme it tracks accuracy cleanly. On ARC, then, the wide model is more accurate but

its halting head is less well calibrated. Near-saturated training accuracy and an almost-constant halting target make this unsurprising, though q deserves care as a ranking signal (Appendix E).

6.3 THE WIDTH MARGIN GROWS WITH THE STEP BUDGET AT LARGE k

The halting statistics above are training-side; a direct held-out test fixes the final checkpoint and varies the step budget at evaluation. Figure 7 reports the resulting margins on ARC-AGI-1 for budgets of 1, 4, 8, and 16 steps; full grids, and a finer 1–4-step sweep, are in Appendix D. The margin is positive in all 24 budget/k cells of the sweep, and in all 24 of the finer one, ranging from $+5.6$ to $+8.8$ points. The pass@1 margin is largest at the smallest budget: with a single step the wide model's pass@1 is 45.1%, above the narrow model at any budget (at most 38.9%). The trend of the margin steepens with k: between budgets 1 and 16 the margin changes by -0.6, -0.3, $+0.1$, $+1.1$, $+1.9$, and $+2.1$ points at $k = 1, 2, 5, 10, 100, 1000$ respectively, and at pass@100 the margin rises monotonically. In the raw pools, TRM's pass@100 and pass@1000 stop improving after four steps, while the wide model's are still rising at sixteen. Beyond four steps, additional iterations no longer improve the narrow model's pool but still improve the wide model's. This is consistent with Section 6.2: for both models the top-1 answer settles in the first supervision step; later steps add pool diversity, and only for the wide model.

6.4 ABLATIONS

Table 7 (Appendix G) summarizes two non-canonical runs at matched ensemble size, both incomplete. First, we tried *pseudo-tokens* as an alternative way to add capacity: each token's embedding is replicated into additional sequence positions, so the recursion runs over a longer sequence rather than a wider residual stream. The results are not as good: at matched evaluations this variant is 1.4 pass@2 points below the GHC wiring, though still 1.5 points above the TRM, so the extra capacity helps but the sequence-length route is the weaker one. Second, recursion is still necessary: a LoopViT-style variant (Shu et al., 2026) without VWN sits 14.4 points below the matched TRM at 6 evaluations.

7 DISCUSSION AND CONCLUSION

Why width matters here. A weight-tied reasoner has no later layers, KV-cache, or chain of thought in which to store partial results; *everything* it knows mid-computation lives in $\boldsymbol{z}_H$ and $\boldsymbol{z}_L$ (Merrill & Sabharwal, 2024; Sapunov, 2026). By the capacity bound of Dambre et al. (2012), a 512-dimensional residual stream carries on the order of 512 independent quantities, so once no free directions remain the loss-minimizing update approaches the identity and the recursion settles. Each of the $2H(L{+}1)$ sublayer applications rewrites that state, so the width bottleneck is paid dozens of times per supervision step rather than once per layer (Seed et al., 2025). This is why a 5.5% widening changes accuracy as much as it does, and why the wide model halts sooner (Sapunov, 2026).

Conclusion. Scaling tiny recursive models has meant scaling depth, but that depth is poorly used: more of it does not help on ARC-AGI-1, and halting settles at one to two of sixteen steps. A 5.5%-parameter widening changes both ends of iteration: one wide supervision step beats sixteen narrow ones, and later steps stay useful after the narrow model's have stopped. The width of the carried state determines how much of a TRM's depth is used.

AI USE STATEMENT

In this work, we have not used generative AI tools to generate synthetic data, to develop the theoretical or conceptual framework, to propose or refine hypotheses, to design or give feedback on the methodology or experiments, to implement VWN+TRM or the training and evaluation code, to formulate or prove mathematical claims, or to interpret results; translation, dataset cleaning, and qualitative data analysis are not applicable to this work. Additionally, we used generative AI tools (a large language model assistant) to edit the paper for readability and to create and edit the plotting and table-generation scripts. We have reviewed all AI-assisted work: every reported number was checked against the metric books and scoring receipts of the versioned metric package described in the reproducibility statement, and all AI-generated plotting and table code was verified against the

source data by the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

Reproducibility statement

All results derive from a versioned metric package containing the six canonical run logs (complete, exit code 0, ten evaluation rows each), per-checkpoint evaluation CSVs, per-run configurations, ARC scoring receipts independently re-derived from the retained `submission.json` files, and SHA-256 manifests over every artifact. Section 5.1 and Appendices A–F specify evaluator semantics, denominators, exact hyperparameters, and hardware; Appendix B reproduces the parameter accounting symbolically. Code for VWN+TRM, dataset construction, and training will be released with the camera-ready version.

References

Cem Anil, Ashwini Pokle, Kaiqu Liang, Johannes Treutlein, Yuhuai Wu, Shaojie Bai, J. Zico Kolter, and Roger B. Grosse. Path independent equilibrium models can better exploit test-time computation. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

Sangmin Bae, Yujin Kim, Reza Bayat, Sungnyun Kim, Jiyoun Ha, Tal Schuster, Adam Fisch, Hrayr Harutyunyan, Ziwei Ji, Aaron Courville, and Se-Young Yun. Mixture-of-recursions: Learning dynamic recursive depths for adaptive token-level computation. In *Advances in Neural Information Processing Systems*, 2025.

Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. Deep equilibrium models. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

François Chollet. On the measure of intelligence. *arXiv preprint arXiv:1911.01547*, 2019.

François Chollet, Mike Knoop, Gregory Kamradt, Bryan Landers, and Henry Pinkard. ARC-AGI-2: A new challenge for frontier AI reasoning systems. *arXiv preprint arXiv:2505.11831*, 2025.

Joni Dambre, David Verstraeten, Benjamin Schrauwen, and Serge Massar. Information processing capacity of dynamical systems. *Scientific Reports*, 2:514, 2012.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers. In *International Conference on Learning Representations*, 2019.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, et al. A mathematical framework for transformer circuits. Transformer Circuits Thread, `https://transformer-circuits.pub/2021/framework/index.html`, 2021.

Katie E. 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. In *International Conference on Machine Learning*, pp. 12666–12700, 2024.

Zitian Gao, Lynx Chen, Yihao Xiao, He Xing, Ran Tao, Haoming Luo, Joey Zhou, and Bryan Dai. Universal reasoning model. *arXiv preprint arXiv:2512.14693*, 2025.

Jonas Geiping, Sean McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian R. Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. Scaling up test-time compute with latent reasoning: A recurrent depth approach. In *Advances in Neural Information Processing Systems*, 2025.

Dobrik Georgiev, Joseph Wilson, Davide Buffelli, and Pietro Liò. Deep equilibrium algorithmic reasoning. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Angeliki Giannou, Shashank Rajput, Jy-yong Sohn, Kangwook Lee, Jason D. Lee, and Dimitris Papailiopoulos. Looped transformers as programmable computers. In *International Conference on Machine Learning*, pp. 11398–11442, 2023.

Alex Graves. Adaptive computation time for recurrent neural networks. *arXiv preprint arXiv:1603.08983*, 2016.

Benhao Huang, Zhengyang Geng, and J. Zico Kolter. Equilibrium reasoners: Learning attractors enables scalable reasoning. In *International Conference on Machine Learning*, 2026.

Alexia Jolicoeur-Martineau. Less is more: Recursive reasoning with tiny networks. *arXiv preprint arXiv:2510.04871*, 2025.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

William Merrill and Ashish Sabharwal. The expressive power of transformers with chain of thought. In *International Conference on Learning Representations*, 2024.

Boris T. Polyak and Anatoli B. Juditsky. Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 30(4):838–855, 1992.

Lucas Prieto, Melih Barsbey, Pedro A. M. Mediano, and Tolga Birdal. Grokking at the edge of numerical stability. In *International Conference on Learning Representations*, 2025.

Grigory Sapunov. Universal transformers need memory: Depth-state trade-offs in adaptive recursive reasoning. *arXiv preprint arXiv:2604.21999*, 2026.

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*, 2025.

Avi Schwarzschild, Eitan Borgnia, Arjun Gupta, Furong Huang, Uzi Vishkin, Micah Goldblum, and Tom Goldstein. Can you learn an algorithm? generalizing from easy to hard problems with recurrent networks. In *Advances in Neural Information Processing Systems*, volume 34, pp. 6695–6706, 2021.

Seed, Baisheng Li, Banggu Wu, et al. Virtual width networks. *arXiv preprint arXiv:2511.11238*, 2025.

Noam Shazeer. GLU variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

Wen-Jie Shu, Xuerui Qiu, Rui-Jie Zhu, Harold Haodong Chen, Yexin Liu, and Harry Yang. LoopViT: Scaling visual ARC with looped transformers. *arXiv preprint arXiv:2602.02156*, 2026.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. RoFormer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.

Guan Wang, Jin Li, Yuhao Sun, Xing Chen, Changling Liu, Yue Wu, Meng Lu, Sen Song, and Yasin Abbasi Yadkori. Hierarchical reasoning model. *arXiv preprint arXiv:2506.21734*, 2025.

Yuxin Wu and Kaiming He. Group normalization. In *European Conference on Computer Vision*, pp. 3–19, 2018.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tie-Yan Liu. On layer normalization in the transformer architecture. In *International Conference on Machine Learning*, pp. 10524–10533, 2020.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Defa Zhu, Hongzhi Huang, Zihao Huang, Yutao Zeng, Yunyao Mao, Banggu Wu, Qiyang Min, and Xun Zhou. Hyper-connections. In *International Conference on Learning Representations*, 2025.

A Evaluation semantics and scoring receipts

Cumulative vote-ensemble pass@k. The ARC evaluator (i) inverse-transforms every augmented prediction to the original grid orientation and crops padding; (ii) hash-deduplicates candidate grids per test output; (iii) records each candidate's vote count and mean $\sigma(q_{\text{halt}})$; (iv) sorts candidates by (vote count, mean q) descending; (v) scores pass@k as gold-among-top k unique candidates, averaged over a task's test outputs, then macro-averaged over tasks; and (vi) does *not* clear its candidate pool between the ten training-time evaluations, so evaluation i reports the cumulative ensemble over checkpoints $1..i$. All final ARC numbers in the paper are therefore cumulative ensembles over the ten evaluation checkpoints for both models. Persisted `submission.json` files retain the top two candidates (`submission_K`=2); pass@$\{5, 10, 100, 1000\}$ use the in-memory pool.

Independently re-derived receipts. Rescoring the retained final submissions against the held-out solutions reproduces the logged top-2 metrics exactly. ARC-AGI-1 (400 tasks / 419 test outputs): TRM pass@1 $= 151.0/400$ task-equivalents ($158/419$ raw grids), pass@2 $= 181.0/400$ ($192/419$); VWN+TRM pass@1 $= 187.0/400$ ($196/419$), pass@2 $= 209.5/400$ ($221/419$). ARC-AGI-2 (120 tasks / 172 outputs): TRM pass@1 $= 5.333/120$ ($8/172$), pass@2 $= 6.333/120$ ($9/172$); VWN+TRM pass@1 $= 10.333/120$ ($15/172$), pass@2 $= 10.833/120$ ($16/172$). Fractional task-equivalents arise from tasks with multiple test outputs.

Held-out example counts. Augmented evaluation rows: 385,815 (ARC-AGI-1), 118,766 (ARC-AGI-2), 422,786 (Sudoku-Extreme). Evaluation always runs all 16 supervision steps (halting disabled), so evaluation compute is identical for both models in a pair.

B Parameter accounting

With $D = 512$, $m = 8$, $n = 40$, $d = 64$, $D' = 2560$, block size $b = 32$, and two transformer layers:

Component	Formula	Params
Token-embedding expansion $E_{D\to D'}$	$b\,D'$	81,920
Puzzle-embedding expansion $E_{D\to D'}$	$b\,D'$	81,920
Answer-head reduce $R_{D'\to D}$ + GroupNorm	$b\,D' + 2D'$	87,040
Halting-head reduce $R_{D'\to D}$ + GroupNorm	$b\,D' + 2D'$	87,040
Dynamic GHC $\times$ 4 (attn+FFN per layer)	$4\,[2mn + n^2 + n(m{+}n) + d(2m{+}n)]$	32,256
GroupNorm inside first attention GHC	$2D'$	5,120
Total ΔP		**375,296**

Each GHC comprises $\boldsymbol{B} \in \mathbb{R}^{m\times n}$, $\boldsymbol{U} \in \mathbb{R}^{n\times m}$, $\boldsymbol{M} \in \mathbb{R}^{n\times n}$, dynamic projections $\boldsymbol{W}_\beta \in \mathbb{R}^{d\times m}$, $\boldsymbol{W}_\alpha \in \mathbb{R}^{d\times(m+n)}$, and gates $\boldsymbol{S}_\beta \in \mathbb{R}^{m\times n}$, $\boldsymbol{S}_\alpha \in \mathbb{R}^{n\times(m+n)}$.

Checkpoint-verified totals: TRM 6,829,058 vs. VWN+TRM 7,204,354 on ARC ($+5.496\%$); 6,828,034 vs. 7,203,330 on Sudoku-Extreme (smaller vocabulary). The block-diagonal maps use $D/b = 16$ input blocks (expansion) or output blocks (reduction) of shape 160×32, giving the $b\,D'$ closed form. The mixing matrix $\boldsymbol{M}$ is identity-initialized and $\boldsymbol{B}, \boldsymbol{U}$ are initialized to the cyclic gather / $[\boldsymbol{I}_m; \boldsymbol{0}]$ pattern of Seed et al. (2025), so the wide model matches the narrow model's function at initialization up to idle lanes.

C Per-checkpoint ARC trajectories

D Held-out step-budget sweeps on ARC-AGI-1

Both sweeps fix the final EMA checkpoint (step $518{,}068$) of the canonical ARC-AGI-1 pair and vary only the step budget (`halt_max_steps`) at evaluation. Each arm uses a fresh dataloader and evaluator over the 400 evaluation tasks with the ranked augmented-pool protocol of Section 3 and cross-checkpoint voting disabled, at global batch 768. Tables 5 and 6 give the full grids. The two sweeps are separate evaluator runs; where they overlap (budgets 1 and 4) values agree to within 0.25 points. Section 6.3 analyzes the margins.

Table 4: Cumulative pass@1/pass@2 (%) at each of the ten evaluation checkpoints (seed 0). VWN+TRM leads at 10/10 ARC-AGI-1 checkpoints and 9/10 ARC-AGI-2 checkpoints (tied pass@1, lower pass@2 at the first ARC-2 evaluation only).

ARC-AGI-1					ARC-AGI-2				
	TRM		VWN+TRM			TRM		VWN+TRM	
Step	p@1	p@2	p@1	p@2	Step	p@1	p@2	p@1	p@2
51,804	20.8	25.8	24.0	26.1	72,381	0.8	1.1	0.8	0.8
103,611	34.1	39.3	35.1	39.8	144,773	0.8	0.8	4.0	4.9
155,419	36.9	41.9	38.0	42.9	217,164	0.8	2.5	4.0	4.4
207,226	37.5	41.4	39.5	44.3	289,555	1.7	2.5	4.0	4.9
259,033	37.5	42.5	42.5	45.8	361,947	2.9	3.2	4.9	6.1
310,840	38.0	42.8	43.9	48.3	434,338	2.9	4.0	5.7	6.9
362,647	37.8	43.3	43.5	48.1	506,729	2.9	4.0	5.7	6.5
414,454	38.3	43.9	44.5	50.5	579,120	3.2	4.4	6.1	6.9
466,261	37.3	44.5	45.4	51.3	651,512	3.2	4.4	7.8	8.6
518,068	37.8	45.3	46.8	52.4	723,904	4.4	5.3	8.6	9.0

Table 5: ARC-AGI-1 step-budget sweep, budgets $\{1, 4, 8, 16\}$ (single final checkpoint; %).

Model	Budget	pass@k (%)					
		k=1	2	5	10	100	1000
TRM	1	36.88	43.75	48.38	51.25	55.38	57.88
	4	38.62	46.25	50.62	52.25	59.00	60.75
	8	38.88	46.75	50.62	53.00	59.25	60.75
	16	38.50	46.75	50.62	52.50	59.00	60.50
VWN+TRM	1	45.12	49.75	54.50	57.00	62.25	63.50
	4	45.62	52.50	56.88	59.12	67.00	68.50
	8	46.12	52.50	57.12	58.62	67.50	68.25
	16	46.12	52.50	56.88	59.38	67.75	68.25

Table 6: ARC-AGI-1 step-budget sweep, finer budgets $\{1, 2, 3, 4\}$ (single final checkpoint; %).

Model	Budget	pass@k (%)					
		k=1	2	5	10	100	1000
TRM	1	36.88	43.75	48.38	51.25	55.38	57.88
	2	38.25	45.38	49.88	52.50	59.25	61.25
	3	38.38	46.00	50.50	52.75	58.75	60.50
	4	38.62	46.25	50.62	52.25	59.00	60.75
VWN+TRM	1	45.12	49.75	54.50	56.75	62.50	63.75
	2	45.50	51.38	56.12	58.12	66.00	66.50
	3	45.62	52.25	56.62	58.88	66.75	68.00
	4	45.62	52.25	56.88	59.12	67.25	68.50

E AUXILIARY HELD-OUT METRICS

Final-evaluation augmented-example metrics (seed 0). ARC-AGI-1 — exact accuracy: TRM 31.29%, VWN+TRM 37.53%; token macro accuracy: 86.55% vs. 88.12%; halting-head accuracy: 74.30% vs. 63.12%. ARC-AGI-2 — exact: 1.81% vs. 3.30%; token: 71.83% vs. 72.78%; halting-head accuracy: 87.76% vs. 82.46%. Sudoku-Extreme — exact: 69.52% vs. 80.70% (matched H3/L6); token: 89.03% vs. 92.65%. The ARC-side divergence between task accuracy (up) and halting-head calibration (down) is discussed at the end of Section 6.2.

F CONFIGURATIONS AND PROVENANCE

Shared. Hidden size 512; 2 post-norm layers (self-attention + SwiGLU, expansion 4, 8 heads); RoPE; puzzle embedding of dimension 512 over 16 prefix positions; stablemax cross-entropy; halting with `no_ACT_continue` (halt if $q > 0$), exploration probability 0.1, $N_{\text{sup}} = 16$; Adam-atan2, $\beta = (0.9, 0.95)$, LR 10^{-4} (constant after 2k warmup), puzzle-embedding LR 10^{-2} (ARC) / 10^{-4} (Sudoku); EMA 0.999; global batch 768; bfloat16; seed 0.

ARC (both benchmarks). H6/L9 ($H = 6$, $L = 9$), $H_{\text{grad}} = 2$; weight decay 0.1; 100k epochs with evaluation every 10k (10 evaluations); ARC-AGI-1: 518,068 optimizer steps (final), 16 GPUs (TRM) vs. 32 (VWN); ARC-AGI-2: 723,904 training steps, same split. VWN: $m = 8$, $n = 40$, dynamic GHC, no activation checkpointing of inner cycles.

Sudoku-Extreme. H3/L6 ($H = 3$, $L = 6$), $H_{\text{grad}} = 1$ (matched pair); weight decay 1.0; 50k epochs, evaluation every 5k; 65,100 training steps; 1 GPU (TRM) vs. 8 (VWN runs).

All six canonical runs completed with exit code 0; offline W&B run IDs: `hg2vsz7w`/`ap1p1kbk` (ARC-1 TRM/VWN), `pgbjmoiy`/`dsg1v3hi` (ARC-2), `8j7q3uyd`/`zigd61w8` (Sudoku TRM/VWN). Ablation runs are partial and identified in Appendix G.

G Extended ablation records

Variant	Evals	p@1	p@2	Δ vs VWN	Δ vs TRM
Alternative realization of width					
pseudo-tokens	4	39.25	42.88	−1.38	+1.50
No virtual width					
LoopViT (no VWN)	6	24.38	28.38	−19.88	−14.38

Table 7: **ARC-AGI-1 ablations** (%; partial runs). *Evals*: evaluations completed (one per 51,807 training steps); p@1, p@2: cumulative pass@1, pass@2; Δ: p@2 minus the canonical VWN+TRM or TRM value at that evaluation count. Row 1: the pseudo-token alternative of Section 6.4; row 2: LoopViT-style recursion without virtual width.

All non-canonical runs are partial (terminated before the full schedule) or evaluation-only; Table 7 compares each family's last surviving cumulative row against the canonical runs at the same evaluation count. Additional recovered families not in the main text: a LoopViT-style "TRM 2×4" variant reached 15.00/18.50 pass@1/2 at one evaluation (vs. 20.75/25.75 canonical TRM); non-looped TRM-ViT baselines reached 2.88/4.88 (one evaluation) and 10.62/13.00 (resumed, two-evaluation checkpoint), confirming that removing recursion is far more damaging than the pseudo-token alternative. The pseudo-token family also has earlier checkpoints consistent with the trend reported; a pre-Triton VWN H3/L4 launch retained configuration receipts but no final metric and is excluded.