SCALING LAWS FOR LATENT REASONING: HOW TEST-TIME COMPUTE TRACKS CIRCUIT DEPTH

Anonymous authors
Paper under double-blind review

ABSTRACT

A recurrent-depth language model spends test-time compute by iterating a weight-shared block T times, reasoning in latent space rather than emitting tokens. We show that this depth axis obeys a mechanistic scaling law: latent accuracy is the cumulative distribution function of a task's circuit depth, discounted by a per-step reliability that grows with model size N. The law makes three falsifiable predictions: (i) accuracy saturates in $\log T$, (ii) the iterations required grow linearly with reasoning depth, and (iii) the value of additional computation is proportional to the serial depth it advances. We verify these predictions on a compute-optimal ladder trained from scratch (19M–121M non-embedding parameters) across three reasoning domains. Fitting the law on smaller models predicts the entire depth-accuracy curve of the held-out largest model (R^2=0.93), with the deeper of two depths requiring more iterations in 461 of 465 untied comparisons. The released Huginn-3.5B follows the same saturating law 29× beyond our ladder on GSM8K, ProsQA, and SVAMP ($R^2 \geq 0.99$), while GSM8K accuracy decays geometrically with gold-solution depth (R^2=0.96). Continuous-thought positions provide substantially less accuracy gain per unit of compute than recurrent depth, and Coconut does not out-scale CoT on public ProsQA. Optimizing the law yields an inference-time analogue of the Chinchilla rule for allocating compute between model size and reasoning depth. Across these settings, test-time compute scales when it advances the serial depth required by the task, establishing circuit depth as the relevant unit of latent computation.

1 INTRODUCTION

Scaling test-time compute has become as important as scaling parameters (Snell et al., 2025; Muennighoff et al., 2025). The dominant approach spends this compute on tokens: generating longer chains of thought or sampling multiple chains and aggregating them (Wei et al., 2022; Wang et al., 2023; Brown et al., 2024). Models can also reason in continuous latent space, spending compute that is invisible in the output text. Recurrent-depth transformers (Geiping et al., 2025) iterate a weight-shared core an arbitrary number of times, adding compute along network depth, while Coconut (Hao et al., 2024) feeds the last hidden state back as the next input embedding, adding steps along the sequence. Yet unlike the token axis, which admits compute-optimal policies and predictive scaling laws (Snell et al., 2025; Wu et al., 2024), the latent axis lacks a comparable account: *if a model can iterate its core T times on a problem, how should accuracy scale with T, and how should a fixed inference budget be split between model size and latent depth?* It remains unknown whether latent accuracy follows a scaling law at all, whether that law transfers across model scale and task distributions, and which latent computation mechanisms actually benefit from additional compute.

We show that a single quantity, the task's *circuit depth* (its longest chain of dependent operations), resolves these questions (Figure 1). A latent iteration contributes to accuracy only insofar as it advances serial reasoning depth. Accuracy as a function of iteration count T is therefore the cumulative distribution function (CDF) of the task's circuit-depth distribution, discounted by a per-step reliability that grows with model size N. This law makes three falsifiable predictions: accuracy saturates in $\log T$; for a task of depth K, the required iterations grow linearly, $T^*(K) \propto K$, at approximately one iteration per reasoning step; and different forms of latent compute should be valuable in proportion to the serial depth they advance. The same principle also connects latent reasoning to CoT: because a reasoning chain externalizes the same serial computation in tokens, CoT follows the same

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

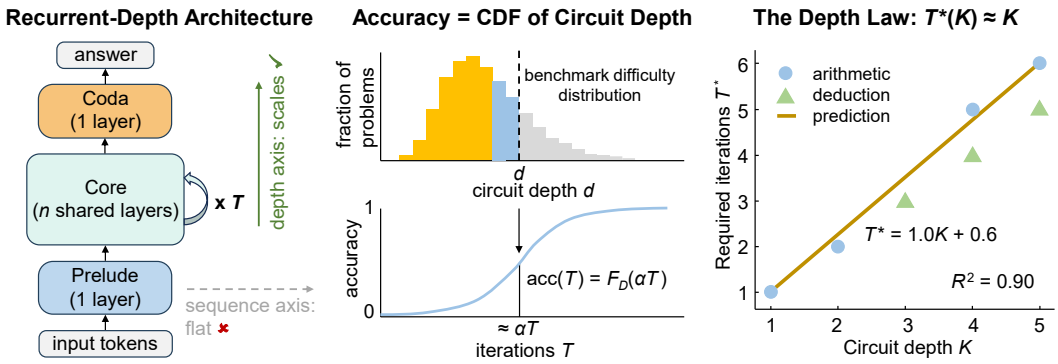


Figure 1: **Latent test-time compute tracks serial circuit depth. (a)** A recurrent-depth model applies a weight-shared core T times, increasing serial depth while keeping sequence length fixed. **(b)** Our theory predicts $\mathrm{acc}(T) = F_D(\alpha T)$: accuracy is the CDF of the benchmark's circuit-depth distribution evaluated at the effective depth αT, yielding diminishing returns with additional iterations. **(c)** Empirically, the iterations required to solve a task scale linearly with its circuit depth K, with slope ≈ 1 across arithmetic and deductive tasks.

CDF law with the token budget replacing latent iterations. Conversely, a latent mechanism that adds sequence positions without advancing comparable serial depth should return less accuracy per unit of compute. Thus, the theory predicts not only *how* latent accuracy scales, but also *which axis of latent compute should scale*.

We test these predictions on a compute-optimal ladder of recurrent-depth transformers trained from scratch (19–121M non-embedding parameters) across three reasoning domains with tunable circuit depth K: modular arithmetic, permutation composition, and symbolic deduction. Controlled tasks are essential because testing a law that relates accuracy to circuit depth requires varying depth while holding task family and format fixed. Natural benchmarks do not provide this control because their computational depth is neither prescribed nor directly observable. Synthetic generators provide the required control while retaining enough accuracy headroom to expose scaling rather than a capacity floor. We then test whether the resulting law transfers beyond this controlled setting using the released Huginn-3.5B and public reasoning benchmarks (§5.3). Our contributions are:

- We derive a mechanistic scaling law for latent reasoning: assuming only that each iteration advances serial circuit depth, accuracy is the cumulative distribution function of the task's circuit-depth distribution, discounted by a model-dependent per-step reliability (§3).
- We show that the law is predictive across model scale: fitting the joint $\mathrm{acc}(N, T)$ law on only the smaller models predicts the held-out largest model's entire depth-accuracy curve (R^2=0.93), model selection rejects the unbounded log-linear form on all 54 measured curves, and the released Huginn-3.5B follows the same functional form far beyond the training ladder (§5.1, §5.3).
- We establish that the recurrent compute required by a problem grows linearly with its circuit depth, at approximately one recurrence per reasoning step. When nominal depth diverges from effective serial depth, the relation breaks, and re-indexing by effective depth restores it, showing that the law tracks the computation itself rather than a task's surface parameterization (§5.2).
- We show that token and latent depth lie on a common compute scale: CoT accuracy follows the chain-length CDF, while recurrent depth provides greater value when partial serial computation is useful. Optimizing the fitted law then yields a Chinchilla-style rule for jointly allocating inference compute between model size and latent reasoning depth (§5.4).

2 Related Work

Latent and recurrent-depth reasoning. Reasoning in hidden states rather than emitted text provides an alternative to token-space computation (Zhu et al., 2025). Recurrent latent computation

traces to cross-layer sharing (Lan et al., 2019), Universal Transformers (Dehghani et al., 2019), and adaptive computation time (Graves, 2016; Banino et al., 2021), and has since been explored in recursive and looped transformers (Bae et al., 2025; Giannou et al., 2023; Saunshi et al., 2025) and recurrent-depth models such as Huginn (Geiping et al., 2025). Other approaches feed hidden states back as embeddings (Hao et al., 2024), chain continuous thoughts (Liu et al., 2025), recursively apply middle layers (Koishekenov et al., 2025), aggregate latent trajectories (You et al., 2025), or learn adaptive halting (Graves, 2016; Banino et al., 2021). Related work internalizes chains of thought (Deng et al., 2023; 2024), learns rationales (Zelikman et al., 2024), or introduces filler tokens (Goyal et al., 2024; Pfau et al., 2024). These works establish latent computation as a viable reasoning substrate, but leave its scaling behavior with model size and depth uncharacterized. Huginn also shows limited decodable latent CoT (Lu et al., 2025), whereas our analysis is behavioral (§K).

Scaling laws for depth-recurrent models. Schwethelm et al. (2026) study the *training-time* value of recurrence through an iso-depth exponent relating loop iterations to equivalent parameters for pretraining loss. We instead study the *inference-time* value of recurrent depth for reasoning accuracy. More broadly, scaling laws relate performance to parameters, data, and compute (Kaplan et al., 2020; Hoffmann et al., 2022), with recent work extending this view to test-time compute (Snell et al., 2025), compute-optimal inference (Wu et al., 2024), and precision (Kumar et al., 2024). We add recurrent latent depth as an explicit test-time scaling axis and derive a corresponding model-size–depth allocation law.

Token-axis test-time scaling. CoT (Wei et al., 2022; Kojima et al., 2022), Self-Consistency (Wang et al., 2023), Best-of-N (Lightman et al., 2023), solution search (Yao et al., 2023), and reinforcement learning over long reasoning traces (DeepSeek-AI, 2025) spend test-time compute along the token axis. Token-level compute can rival parameter scaling (Snell et al., 2025; Muennighoff et al., 2025), but gains concentrate on math and symbolic reasoning (Sprague et al., 2024), saturate under repeated sampling (Brown et al., 2024), and unverified scaling can be suboptimal (Setlur et al., 2025). We complement this work by treating recurrent depth as a distinct compute axis and comparing the two under matched FLOPs. Their scaling differs qualitatively: recurrent depth saturates with iteration count, whereas CoT follows the log-linear regime predicted for token-based computation.

Circuit complexity of reasoning. A fixed-depth transformer with logarithmic precision lies in the parallel complexity class TC^0 (Merrill & Sabharwal, 2023), motivating serial computation for inherently sequential problems. Chain-of-thought provides such computation through successive reasoning steps (Merrill & Sabharwal, 2024; Feng et al., 2023; Li et al., 2024), while the globality barrier gives a concrete example of a limitation overcome by scratchpads (Abbe et al., 2024). State tracking and permutation composition provide canonical serial reasoning tasks (Merrill et al., 2024). These results motivate our central precondition: latent scaling requires iterations to advance serial circuit depth. Under this precondition, our theory predicts both a saturating depth-accuracy curve and a linear depth requirement, $T^*(K) \propto K$. It also explains why latent sequence computation that advances little serial depth can scale much less effectively (Claim 3).

3 Circuit-Depth Theory of Latent Scaling

Setup and notation. A recurrent-depth model consists of a prelude of l_P layers that embeds the input, a core of l_R weight-shared layers applied T times, and a coda of l_C layers that decodes the final state (Geiping et al., 2025). Let the core be the map $g_\theta(\cdot\,; e) : \mathbb{R}^h \to \mathbb{R}^h$, which updates a latent state given the fixed input embedding e. The model "thinks" by iterating

$$h_{t+1} = g_\theta(h_t; e), \qquad t = 1, \ldots, T, \tag{1}$$

after which the coda decodes h_T. We separate two factors that determine the value of these iterations: how much circuit depth they cover and how reliably the model executes each reasoning step.

Definition 1 (Circuit depth)**.** *The* circuit depth $d(x)$ *of a problem x is the minimum serial depth of a circuit that computes its answer, equivalently the length of its longest chain of dependent operations. In the controlled tasks used to fit the depth law, $d(x) = K$ by construction, with each reasoning step depending on the preceding one.*

Definition 2 (Per-step reliability and resolution rate)**.** *Let $p(N) \in (0, 1]$ be the* per-step reliability*, the probability that a size-N model executes one reasoning step correctly. Let $\alpha > 0$ be the* resolution rate*, the number of reasoning steps advanced per core iteration. Thus T iterations can resolve*

circuit depth up to $\lfloor \alpha T \rfloor$. The two quantities play distinct roles: $p(N)$ determines the accuracy ceiling, while α determines how rapidly depth is covered.

Proposition 1 (Accuracy is a circuit-depth CDF)**.** *Under Definitions 1–2 with independent per-step execution, the accuracy of a recurrent-depth model on a benchmark with circuit-depth distribution $\mathcal{D}$ is*

$$\mathrm{acc}(N,T) \;=\; \mathbb{E}_{d\sim\mathcal{D}}\big[\mathbf{1}[d \le \alpha T]\, p(N)^{\,d}\big] \;\approx\; \underbrace{F_{\mathcal{D}}(\alpha T)}_{\text{depth covered}} \cdot \underbrace{p(N)^{\,\bar{d}}}_{\text{steps executed}}\,, \tag{2}$$

where $F_{\mathcal{D}}$ is the CDF of $\mathcal{D}$ and $\bar{d} = \mathbb{E}[d \mid d \le \alpha T]$. The exact form retains the depth-dependent reliability inside the expectation, while the approximation factors it at the conditional-mean depth $\bar{d}$.

A problem is solved when T iterations cover its circuit depth ($d \le \alpha T$) and all d dependent steps execute correctly, which occurs with probability $p(N)^d$ under independence. Averaging over $d \sim \mathcal{D}$ gives the exact expression in Equation 2. We then obtain the approximation by separating the two mechanisms: $F_{\mathcal{D}}(\alpha T)$ captures depth coverage, while $p(N)^{\bar{d}}$ captures execution reliability. The approximation is exact for the fixed-depth controlled tasks ($\mathcal{D}$ a point mass) and becomes tight when $p(N) \to 1$ or the covered-depth distribution is concentrated. Its error is $O\big((\log p(N))^2\,\mathrm{Var}[d \mid d \le \alpha T]\big)$. We quantify this error and relax the independence and constant-rate assumptions in §A. Crucially, the saturating dependence on $\log T$ follows from the coverage term $F_{\mathcal{D}}(\alpha T)$ and does not depend on the factorization. Two falsifiable predictions follow directly.

Claim 1 (Saturating form)**.** *As a function of $\log T$, $\mathrm{acc}(N,T)$ is a bounded, monotone curve governed by the CDF $F_{\mathcal{D}}$, with its asymptotic ceiling set by model size through $p(N)$. CoT covers depth by emitting steps. Its length-coverage is also a CDF, paced by the per-step token cost τ (Proposition 2), while scaling test-time compute by sampling more chains produces log-linear growth toward a depth-bounded ceiling. These distinct functional forms can be distinguished by model selection.*

Claim 2 (Linear depth-law)**.** *On a task of fixed circuit depth K, $F_{\mathcal{D}}$ is a step at K. Accuracy therefore turns on when $\alpha T \ge K$: the onset $T^*(K) = K/\alpha + c$ is linear in K, with slope $1/\alpha$ iterations per reasoning step.*

We find empirically that the fitted slope $1/\alpha$ decreases with model size (§5.1). A natural interpretation is that a less reliable executor must re-derive failed steps, reducing its effective resolution rate. Because Equation 2 treats $p(N)$ and α as independent parameters, we interpret this coupling as an empirical regularity rather than a derived consequence.

Implementation-agnostic prediction. Proposition 1 depends only on whether an iteration advances bounded serial depth, not on how that iteration is implemented.

Claim 3 (Implementation-agnostic form)**.** *Any latent-compute scheme whose iterations advance a bounded amount of serial reasoning satisfies Equation 2 with its own resolution rate α. This includes recurrent shared-core iterations and continuous thoughts appended along the sequence axis, as in Coconut. Both therefore exhibit saturating accuracy in iteration count, differing only in α.*

The resolution rate therefore determines how effectively a latent-compute mechanism converts iterations into serial reasoning depth: a scheme that advances little depth per iteration should scale more slowly even if its accuracy follows the same saturating form. We test this prediction directly in §5.2, where the continuous-thought axis returns a small fraction of the gain obtained from recurrent depth on matched models. CoT provides the complementary case by externalizing each reasoning step as a token.

Proposition 2 (Token-axis coverage)**.** *Let τ be the* verbalization cost*, the tokens needed to write one reasoning step. A chain of L generated tokens covers depth L/τ, so*

$$\mathrm{acc}_{\mathrm{CoT}}(N,L) = \mathbb{E}_{d\sim\mathcal{D}}\big[\mathbf{1}[d \le L/\tau]\, p(N)^{d}\big]\,,$$

a covered-depth CDF in L rather than T. Aggregating M independent chains gives coverage $1-(1-q)^M$ in the per-chain success probability q, which grows log-linearly in M and is bounded by the depth ceiling of a single chain.

The two axes therefore scale for different reasons. Latent iterations extend the depth that a single computation can resolve, whereas additional CoT samples repeatedly probe a depth-limited computation. The former yields CDF-like saturation through depth coverage, while the latter exhibits diminishing returns toward a depth-bounded ceiling. The token cost τ further makes their relative efficiency task-dependent.

Scaling law for latent reasoning. We now combine the shape (Claim 1) and onset (Claim 2) predictions. Using a logistic covered-depth CDF in Proposition 1 gives the closed-form law we fit and test throughout:

$$\mathrm{acc}(N,T) = \underbrace{a_0 N^{a_A}}_{A(N)\approx p(N)^{\bar{d}}} \cdot \underbrace{\frac{1}{1+\exp(-k(\log T - \log T_0(N)))}}_{F_{\mathcal{D}}(\alpha T)}. \tag{3}$$

Here $A(N) = a_0 N^{a_A}$ is the accuracy ceiling and the onset $\log T_0(N) = \log t_0 - a_T \log N$ satisfies $\log T_0 \approx \log(K/\alpha)$ (Claim 2). The steepness k is set by the width of $\mathcal{D}$, sharper for a more concentrated depth distribution, and the ceiling exponent a_A carries the size scaling of $p(N)$ (§G). The power-law ceiling is a local description: it is unbounded above, reaching 1 at 81M and exceeding the measured 121M ceiling (0.99) by 0.10. Reading $p(N)$ off $A(N)$ is quantitative only within the fitted range, and extending it requires the bounded ceiling tested in §F.

4 Experimental Setup

Model ladder. Following the Huginn recipe (Geiping et al., 2025), we train recurrent-depth transformers from scratch at six sizes ($19, 30, 44, 59, 78, 121$M non-embedding parameters), each with a size-matched, compute-optimal pretraining budget (Hoffmann et al., 2022). This control isolates the effect of model size N: a fixed token budget can undertrain larger models and even reverse the expected size ordering (§F). We give architecture and pretraining details in §B.

Controlled tasks. We construct three synthetic reasoning domains with tunable depth and low chance accuracy: **StateTrack** (modular arithmetic), **PermTrack** (permutation composition), and **LogicTrack** (symbolic deduction). Each composes a chain of K dependent steps, so solving an instance requires serial computation across its depth. The domains differ in what each step computes: a modular update, a swap of two slots, or a rule application. We additionally construct **TreeTrack**, a branching task in which circuit depth is the tree height h while operation count grows as $2^h - 1$, allowing us to separate serial depth from problem length. We fine-tune models on step-by-step targets and evaluate them greedily on held-out problems while varying the recurrent iteration count T. Details of task construction, required depths, and fine-tuning depth ranges are given in §C.

Baselines and external tests. We compare recurrent depth with two alternative ways of spending test-time compute. On the token axis, size-matched standard transformers generate explicit reasoning chains, with compute controlled by the number of generated tokens L. On the sequence axis, we train Coconut-style continuous-thought models on the same tasks and evaluate them with c latent positions (§5.2). Because each continuous thought is fed back as an input embedding, this arm uses the standard-architecture twin of each rung, matched in width, layer count, corpus, and token budget (§I). We then test the fitted law on the released Huginn-3.5B checkpoint and on public benchmarks including GSM8K, ProsQA, and SVAMP. Finally, we evaluate Coconut (Hao et al., 2024) on Pythia-410M–2.8B using ProntoQA and ProsQA. Together, these experiments test whether the observed scaling behavior transfers across model scale, task distribution, and latent computation mechanism. We report FLOP accounting details in §D.

Scaling-law fitting. We fit the proposed law in Equation 3 to the measured (N, T) curves by nonlinear least squares, and compare four candidate forms—sigmoid, exponential saturation, power law, and log-linear—using R^2 and the Akaike information criterion (AIC). We fit all six sizes jointly for the main analysis, and separately hold out the 121M model, fitting only the five smaller models and testing the prediction on the unseen model. We average fine-tuning results across seeds. Full seed counts, confidence intervals, and model-selection statistics are reported in §E and §F.

5 Results

5.1 Latent test-time compute follows a mechanistic scaling law

Does latent test-time compute obey the predicted scaling law? Our first prediction (Claim 1) is that accuracy should improve with latent compute but eventually saturate. This behavior is consistent across the design: on all 54 latent curves, a saturating function of $\log T$ outperforms an unbounded

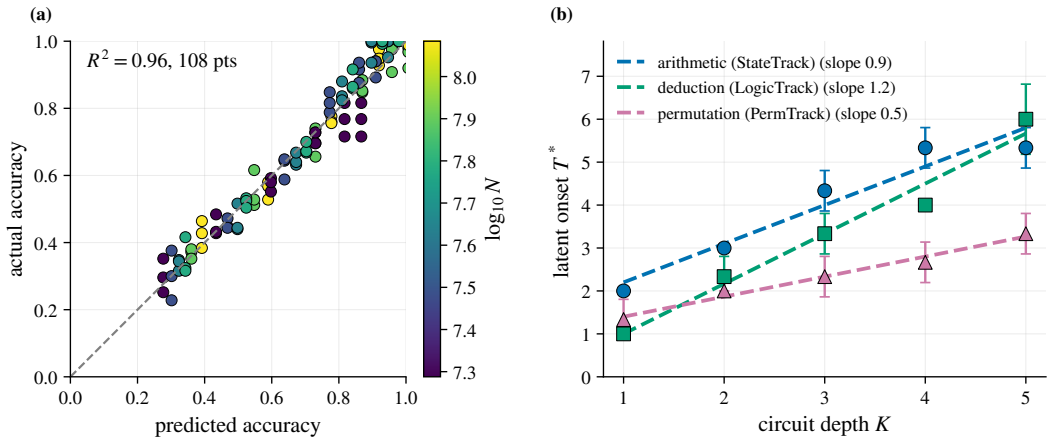


Figure 2: **Cross-scale prediction and depth-law scaling. (a)** Predicted versus actual accuracy at all 108 (N, T) operating points (color: $\log_{10} N$), an in-sample view of the six-size surface (R^2=0.96). The held-out test of the same law is Table 1. **(b)** The latent onset $T^*(K)$ at 44M grows approximately linearly with circuit depth on all three reasoning domains, seed-averaged with ± 1 standard deviation.

Table 1: **Small models predict a larger model's depth-scaling curve.** The joint law $\mathrm{acc}(N, T)$ fit on the five smaller ladder models (19–78M) predicts the held-out 121M model's entire $\mathrm{acc}(T)$ curve at R^2=0.93, MAE 0.045 over its 18 points (§5.1). Rows average the three fine-tuning seeds. Predicted values are clipped to $[0, 1]$ for display: the fitted ceiling is an unbounded power law in N (§3), and the unconstrained fit gives 1.05 and 1.11 at T=16 and 32. The quoted R^2 conservatively uses the unconstrained values, since clipping raises it to 0.98.

Latent depth T	1	2	4	8	16	32
Actual (121M, held out)	0.43	0.56	0.76	0.95	0.99	0.99
Predicted from 19–78M	0.38	0.58	0.78	0.94	1.00	1.00

log-linear form, with R^2=0.94–1.00 and improvements of 0.02–0.30 in R^2, and is preferred by AIC on every curve (§E). The particular saturating parameterization matters little: all three forms perform similarly. The same behavior appears for the released Huginn-3.5B on GSM8K. This is what a coverage account predicts: the relevant quantity is not simply how much latent compute is available, but how much of the task's required computation it covers.

Saturation, however, does not by itself imply a depth mechanism. If each additional recurrence advances serial circuit depth, then the onset of accuracy should move systematically with the depth a problem requires. This is exactly what we observe: deeper problems turn on at larger T across the model ladder and on all three controlled tasks (Figure 2b). Depth also grades GSM8K: the deeper of two gold-solution strata is less accurate in 119 of 146 untied comparisons (Kendall τ=0.61), while their plateau accuracies decrease geometrically with depth (R^2=0.96, §K). Yet all strata turn on within T=8–10, so the released model resolves the depth-dependent ceiling rather than the onset. The controlled tasks provide the complementary view. Together, the two regimes separate the law into its two observable components: the rising part of $\mathrm{acc}(T)$ reflects coverage of the required circuit depth, while the plateau reflects the reliability with which that depth is executed.

Does the law extrapolate across model scale? The more stringent test of a scaling law is whether it predicts behavior that was not used to fit it. We therefore hold out the 121M model and fit the law using only the five smaller models, spanning 19–78M. The resulting fit predicts the *entire* depth-accuracy curve of the unseen 121M model with R^2=0.93 and mean absolute error 0.045 (Table 1, Figure 2a), with comparable error under leave-one-size-out cross-validation. The extrapolation fails when the fit is restricted to the pre-saturation regime ($T \leq 8$), giving $R^2 < 0$ beyond the fitted range because the ceiling is not identifiable before the curve approaches saturation. The agreement across the full curve therefore cannot be explained by fitting the local rise alone: the law captures the joint dependence of accuracy on model size and latent depth well enough to predict the test-time compute curve of a model it has never seen.

How does model size enter the law? Model size affects the scaling curve in two distinct ways: it determines how much serial depth can be reached and how reliably that depth is executed. On $K \leq 5$, the task is already solved by the 59M model, leaving the ceiling exponent weakly identified (a_A=0.10, 95% CI $[0.00, 0.17]$). To expose the size dependence of the ceiling, we instead fine-tune the same models on the wider $K \leq 12$ range, where the ceiling spans 0.67–1.00 and gives a_A=0.22 (95% CI $[0.11, 0.26]$). We can then separate this aggregate effect from execution reliability by stratifying problems by effective serial depth. The plateau at depth d follows A_d=$p(N)^d$, with $p(N)$=0.82–0.96 and $p(N) \propto N^{0.12}$ at the four sizes where p is identifiable (Figure 3, §G). Thus, the increase in accuracy with model size is not only a higher ceiling: larger models execute each additional unit of serial depth more reliably. This reliability trend rises as a power law across the ladder and remains measurable on Huginn-3.5B, 29× larger than our largest training model. Model scale therefore improves both terms of the law: how far the model can reason and how reliably it executes the depth it reaches.

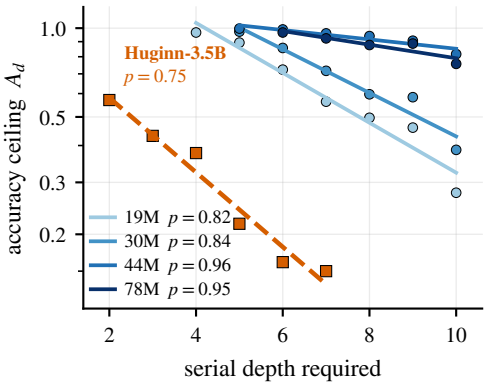


Figure 3: **The reliability term at two scales.** Ceiling A_d against required serial depth, log axis: effective depth on StateTrack, gold reasoning lines on GSM8K. A straight line recovers A_d=$p(N)^d$ (R^2=0.83–0.96, §G).

5.2 Serial circuit depth is the relevant axis of latent computation

Does required latent compute grow with serial depth? The scaling law makes a stronger prediction than saturation (Claim 2): the latent budget required to solve a problem should grow linearly with the serial depth of the computation. We test this prediction by defining $T^*(K)$ as the smallest iteration count at which depth-K accuracy reaches 0.5. Averaging over three seeds, the 44M model gives

$$T^*(K) = 0.90\,K + 1.30, \qquad R^2 = 0.93,$$

with the same relationship on LogicTrack, $1.17\,K-0.17$, and PermTrack, $0.47\,K+0.93$ (Figure 2b). The conclusion does not depend on fitting a linear model: across the 54 curves, the deeper of two depths has the larger T^* in 461 of 465 untied comparisons. We further find that the onset slope moves toward one recurrence per level of depth as models become more reliable, falling from 1.43±0.06 at 19M to 0.87±0.12 at 121M (Table 5). The depth dependence persists on unseen depths $K \in \{6, \ldots, 9\}$, where accuracy continues to rise with T at every size (§G). Thus, within the measured regime, serial depth provides a direct estimate of the latent budget required by a problem: $T \approx K$, up to a task-dependent calibration described in §G.

What happens when nominal depth stops tracking serial depth? If the law is about computation rather than a task's surface parameterization, it should break when nominal depth ceases to measure the depth required to produce the answer. We test this by extending StateTrack fine-tuning from $K \leq 5$ to $K \leq 9$. Nominal-K onsets compress beyond K=4, and the full-range slope falls to 0.43 (R^2=0.79) at 44M and 0.42 (R^2=0.64) at 78M. The reason is that multiplication modulo 10 is non-injective: a suffix of operations can compose to a constant map, making the answer independent of operations earlier in the chain. Consequently, mean effective serial depth falls from $0.91K$ at K=5 to $0.64K$ at K=12. Re-indexing the instances by this label-measured effective depth restores the predicted ordering at all six sizes, with 547 concordant versus 17 discordant depth pairs (Kendall τ=0.85, $p<10^{-100}$), zero discordant pairs at the four largest sizes, and recovered slopes of 0.47–1.98 at R^2=0.67–0.84. The apparent breakdown is not a failure of the depth law, but a failure of nominal depth to track the computation: scaling follows the serial depth an answer requires, and a generator parameter follows that quantity only while its steps preserve information (§G).

Does latent compute track work or critical-path depth? On the three linear-chain tasks, operation count and serial depth grow together, so they cannot distinguish which of the two the latent budget tracks. We therefore turn to TreeTrack, which separates them: evaluating a balanced binary expression tree of height h requires 2^h-1 operations but only h serial steps along its critical path. Accuracy follows the latter. The saturating form holds at all six sizes (sigmoid $R^2 \geq 0.97$ against

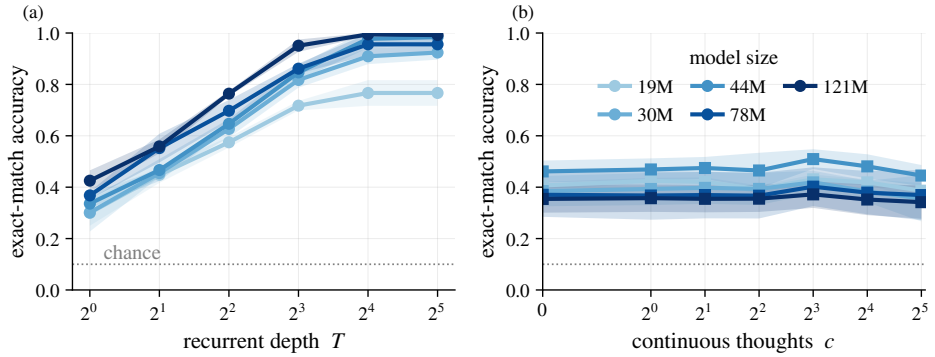


Figure 4: **The two latent axes are not equivalent.** On matched models and tasks (StateTrack at $K \leq 5$), the **(a)** depth axis (recurrent iterations T) produces strong depth-graded scaling, while the **(b)** sequence axis (Coconut-style continuous-thought positions c) gains at most 0.05. What distinguishes them is how much serial depth each additional unit advances (§I).

log-linear 0.72–0.90), height-1 trees saturate by $T{=}4$, and taller trees turn on progressively later and plateau lower. An operation-count interpretation would instead require a budget growing exponentially with h (§H). We conclude that the relevant complexity measure is the critical path through the computation, not the amount of work exposed on its surface.

Does every latent computation advance serial depth equally? If serial depth is the scarce resource, then different ways of spending latent compute should be valuable in proportion to the depth they advance. Continuous thoughts provide a direct test: they add latent sequence positions, but each position performs only a single forward pass rather than another recurrence of the depth-limited core. On matched models and tasks, we find that recurrent iterations improve StateTrack accuracy by 0.47–0.65 across all five sizes, compared with at most 0.05 for continuous thoughts (Figure 4). On the wider depth range, where additional depth matters more, continuous thoughts gain $+0.07$ to $+0.17$, versus $+0.33$ to $+0.62$ for recurrent depth at the same sizes. Their gain is nevertheless depth-graded: at 44M it rises from $+0.08$ on $K \leq 4$ to $+0.23$ on $K \geq 9$, showing that continuous thoughts do advance serial depth, but much more slowly. Per unit of budget, they provide between an eighth and a quarter of the gain of recurrent iterations (§I). We observe the same asymmetry on unseen depths $K{\in}\{6, \ldots, 9\}$, where continuous thoughts gain 0.06–0.12 non-monotonically, compared with 0.34–0.58 for recurrent depth. It also appears for Coconut on standard Pythia models and public ProsQA (§J). Thus, latent compute is not a homogeneous resource: what matters is not the number of latent positions, but the serial depth that those positions make available.

5.3 The law transfers across scale and task distribution

Does the law hold beyond the controlled ladder? We finally test the law on the released Huginn-3.5B, which is 29–180$\times$ larger than the ladder we fit. On GSM8K, ProsQA, and SVAMP, we find the same saturating dependence of latent accuracy on $\log T$, with sigmoid fits at $R^2{\geq}0.99$ against 0.76–0.86 for log-linear. On GSM8K, we stratify all 1319 test problems by the number of reasoning lines in the gold solution and find plateaus that fall geometrically with depth: regressing $\log A_d$ against depth gives $R^2{=}0.96$ and a per-step reliability of $p{=}0.747$, at which the attainable plateau halves every 2.4 reasoning lines. We therefore recover both components of the law, depth-dependent coverage and depth-dependent execution reliability, in a released model far outside the scale and task distribution used to fit it. Figure 3 places its reliability decay on the same axis as our ladder.

The same data also reveals why the two components are separately identifiable in different regimes. The GSM8K strata turn on at similar iteration counts, $T{=}8$–10, consistent with the controlled result that coverage ceases to be the binding constraint when execution reliability is low enough to hold the deeper strata down. We therefore use the controlled tasks to identify the onset-depth relation and the released model to test the depth-dependent reliability term. We can also run the law in reverse: normalized per-iteration accuracy gains estimate how much of a benchmark's depth distribution each budget covers. The recovered distributions place SVAMP shallower than GSM8K, matching the ordering of their gold solutions (§K).

5.4 THE LAW PREDICTS INFERENCE-TIME COMPUTE ALLOCATION

Do depth and tokens define different compute frontiers? We interpret the two axes through a common principle: each budget determines how much serial reasoning can be completed before the computation terminates. Proposition 2 predicts that a chain contributes only when it reaches its terminating answer, making token-budget accuracy the CDF of the chain-length distribution, directly analogous to latent accuracy as the CDF of circuit depth. We verify this correspondence to three decimals at all 11 budgets on StateTrack and PermTrack, and find that chain length is itself linear in reasoning depth at $R^2 \geq 0.997$ across the task families we measure (§D).

This correspondence produces distinct frontiers at matched inference FLOPs (Figure 8 in §J). We find latent depth to be more efficient at low budgets, because partial depth coverage already improves accuracy, whereas a chain contributes only after the required intermediate computation has been externalized. CoT overtakes it once the budget suffices to complete the relevant chains, at 4.1×10^9 FLOPs on StateTrack and 6.3×10^9 on PermTrack. The ordering of these crossovers follows the number of tokens each task spends per reasoning step. The crossover therefore marks where the more economical primitive changes for a given task, rather than a universal compute threshold. Neither axis shows meaningful overthinking within the measured range: the peak-to-final accuracy drop is at most 0.028 across the ladder and all three Huginn benchmarks (§G).

How should inference budget be split between size and depth? We next use the fitted law to turn the scaling relation into an inference-time allocation rule. Under a fixed per-problem FLOP budget C, optimizing the fitted surface gives $N^*(C) \propto C^{0.42}$ and $T^*(C) \propto C^{0.69}$ on the wide-depth ladder (Figure 6), with the optimum moving from 19M at one iteration to 78M at sixteen. We test whether the allocation rule extrapolates: training the law on budgets below $0.6C$ recovers the optimal size on held-out budgets to within one ladder rung. The exponents are identifiable only where the ceiling is, since on the $K \leq 5$ ladder the optimum never leaves the smallest rung. Although the exponents depend on the training regime and ceiling parameterization (§F), the procedure is general: the law converts a measured size-depth scaling surface into a budget-dependent inference policy, allowing us to optimize model size and latent reasoning depth jointly rather than tune them independently.

6 CONCLUSION

We show that latent reasoning depth provides a predictable axis for scaling test-time compute, governed by a mechanistic law: accuracy as a function of recurrent iterations follows the CDF of task circuit depths, with the onset advancing at approximately one recurrence per reasoning step and the attainable ceiling rising with model size. On a compute-optimal ladder trained from scratch across three reasoning domains, we measure these two terms separately: the onsets reveal a resolution rate near one iteration per step, while depth-stratified ceilings reveal a per-step reliability whose predicted geometric decay also appears across all 1319 GSM8K problems for the released Huginn-3.5B (R^2=0.96). We further show that the law is predictive, not merely descriptive: fitting it on smaller models predicts the held-out largest model's entire depth curve, and optimizing the fitted surface yields a Chinchilla-style rule for allocating inference compute between model size and latent depth. The framework also puts different forms of test-time compute on a common scale: on our controlled tasks, one reasoning step costs approximately one recurrence or 14 emitted tokens, while continuous thoughts return only an eighth to a quarter as much accuracy per unit of budget. For depth-structured reasoning, the required latent budget can therefore be estimated directly from circuit depth, $T \approx K$: test-time compute is most effective when it is spent along the axis that advances the serial computation a task requires.

Limitations. Our quantitative law spans a $6\times$ size range (19–121M), while the released Huginn-3.5B validates the functional form at $29\times$ larger scale without allowing us to test the size dependence itself (§L). We calibrate the effective parameters $p(N)$ and α once per task: the resolution rate reflects the computational cost of a step, while a nominal depth parameter tracks circuit depth only when its steps preserve information (§G). We recover the ceiling exponent only on the wide-depth ladder, because it is identifiable when tasks remain capacity-limited (§F). Extending the quantitative scaling law to larger models, broader task families, and less controlled reasoning settings remains an important direction for future work.

AI USE STATEMENT

In this work, we used generative AI tools for designing or providing feedback on research methodology or experiments, implementing methods, and interpreting results. We have not used generative AI tools for generating synthetic data sets, developing theoretical models or conceptual frameworks, formulating mathematical claims, providing critical ingredients for proving mathematical claims, assisting in the writing of proofs, or proposing or refining hypotheses. Assisting with translation, cleaning and reformatting datasets, and supporting qualitative and thematic data analysis are not applicable to this work. Additionally, we used generative AI tools for editing the paper to improve readability and for identifying relevant literature. We have reviewed all AI-assisted work. LLM-generated code was verified and tested for correctness, AI-suggested interpretations were checked against outputs, and AI-identified references were verified against their source records. 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 models are trained from scratch using the recipe in §B, including architecture, optimizer, token budgets, and random seeds. The three controlled tasks are fully specified as generators in §C. FLOP accounting is detailed in §D, and the fitting and model-selection procedures, including the exact functional forms and parameter bounds, are specified in §4 and §E. The Huginn-3.5B anchor uses the publicly available checkpoint. Evaluation prompts are tokenized exactly as in fine-tuning, without an added special token, and each sweep instantiates the generator named by the task it is evaluating: both conventions are asserted in the released sweep code, since a mismatch in either is silent and can move a long-generation baseline from its true accuracy to chance. Held-out problems are drawn from the same generators under a different seed, so at the smallest depths the instance space is small enough that overlap with training is unavoidable by construction, and we report it per depth in §C. These details fully specify the experimental setup and analysis needed to reproduce the reported scaling curves and fitted laws.

REFERENCES

Emmanuel Abbe, Samy Bengio, Aryo Lotfi, Colin Sandon, and Omid Saremi. How far can transformers reason? the globality barrier and inductive scratchpad. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Sangmin Bae, Adam Fisch, Hrayr Harutyunyan, Ziwei Ji, Seungyeon Kim, and Tal Schuster. Relaxed recursive transformers: Effective parameter sharing with layer-wise lora. In *International Conference on Learning Representations (ICLR)*, 2025.

Andrea Banino, Jan Balaguer, and Charles Blundell. Pondernet: Learning to ponder. In *ICML Workshop on Automated Machine Learning*, 2021.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V. Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*, 2024.

DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

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

Yuntian Deng, Kiran Prasad, Roland Fernandez, Paul Smolensky, Vishrav Chaudhary, and Stuart Shieber. Implicit chain of thought reasoning via knowledge distillation. *arXiv preprint arXiv:2311.01460*, 2023.

Yuntian Deng, Yejin Choi, and Stuart Shieber. From explicit cot to implicit cot: Learning to internalize cot step by step. *arXiv preprint arXiv:2405.14838*, 2024.

Guhao Feng, Bohang Zhang, Yuntian Gu, Haotian Ye, Di He, and Liwei Wang. Towards revealing the mystery behind chain of thought: A theoretical perspective. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

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. *arXiv preprint arXiv:2502.05171*, 2025.

Angeliki Giannou, Shashank Rajput, Jy-yong Sohn, Kangwook Lee, Jason D Lee, and Dimitris Papailiopoulos. Looped transformers as programmable computers. *arXiv preprint arXiv:2301.13196*, 2023.

Sachin Goyal, Ziwei Ji, Ankit Singh Rawat, Aditya Krishna Menon, Sanjiv Kumar, and Vaishnavh Nagarajan. Think before you speak: Training language models with pause tokens. In *International Conference on Learning Representations (ICLR)*, 2024.

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

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space. *arXiv preprint arXiv:2412.06769*, 2024.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Yeskendir Koishekenov, Aldo Lipani, and Nicola Cancedda. Encode, think, decode: Scaling test-time reasoning with recursive latent thoughts. *arXiv preprint arXiv:2510.07358*, 2025.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Tanishq Kumar, Zachary Ankner, Benjamin F Spector, Blake Bordelon, Niklas Muennighoff, Mansheej Paul, Cengiz Pehlevan, Christopher Ré, and Aditi Raghunathan. Scaling laws for precision. *arXiv preprint arXiv:2411.04330*, 2024.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2019.

Zhiyuan Li, Hong Liu, Denny Zhou, and Tengyu Ma. Chain of thought empowers transformers to solve inherently serial problems. *arXiv preprint arXiv:2402.12875*, 2024.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.

Jiayu Liu, Zhenya Huang, Xuan Yang, Tianyun Ji, Anya Sims, Hao Xu, Enhong Chen, Yee Whye Teh, and Ning Miao. Deep thinking by markov chain of continuous thoughts. *arXiv preprint arXiv:2509.25020*, 2025.

Wenquan Lu, Yuechuan Yang, Kyle Lee, Yanshu Li, and Enqi Liu. Latent chain-of-thought? decoding the depth-recurrent transformer. *arXiv preprint arXiv:2507.02199*, 2025.

William Merrill and Ashish Sabharwal. The parallelism tradeoff: Limitations of log-precision transformers. *Transactions of the Association for Computational Linguistics*, 11:531–545, 2023.

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

William Merrill, Jackson Petty, and Ashish Sabharwal. The illusion of state in state-space models. *arXiv preprint arXiv:2404.08819*, 2024.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling. *arXiv preprint arXiv:2501.19393*, 2025.

Jacob Pfau, William Merrill, and Samuel R Bowman. Let's think dot by dot: Hidden computation in transformer language models. In *Conference on Language Modeling (COLM)*, 2024.

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 (ICLR)*, 2025.

Kristian Schwethelm, Daniel Rueckert, and Georgios Kaissis. How much is one recurrence worth? iso-depth scaling laws for looped language models. *arXiv preprint arXiv:2604.21106*, 2026.

Amrith Setlur, Nived Rajaraman, Sergey Levine, and Aviral Kumar. Scaling test-time compute without verification or rl is suboptimal. *arXiv preprint arXiv:2502.12118*, 2025.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In *International Conference on Learning Representations*, 2025.

Zayne Sprague, Fangcong Yin, Juan Diego Rodriguez, Dongwei Jiang, Manya Wadhwa, Prasann Singhal, Xinyu Zhao, Xi Ye, Kyle Mahowald, and Greg Durrett. To CoT or not to CoT? chain-of-thought helps mainly on math and symbolic reasoning. *arXiv preprint arXiv:2409.12183*, 2024.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations*, 2023.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, 2022.

Bohong Wu, Mengzhao Chen, Xiang Luo, Shen Yan, Qifan Yu, Fan Xia, Tianqi Zhang, Hongrui Zhan, Zheng Zhong, Xun Zhou, Siyuan Qiao, and Xingyan Bin. Parallel loop transformer for efficient test-time computation scaling. *arXiv preprint arXiv:2510.24824*, 2025.

Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. Inference scaling laws: An empirical analysis of compute-optimal inference for problem-solving with language models. *arXiv preprint arXiv:2408.00724*, 2024.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*, 2023.

Runyang You, Yongqi Li, Meng Liu, Wenjie Wang, Liqiang Nie, and Wenjie Li. Parallel test-time scaling for latent reasoning models. *arXiv preprint arXiv:2510.07745*, 2025.

Eric Zelikman, Georges Harik, Yijia Shao, Varuna Jayasiri, Nick Haber, and Noah D Goodman. Quiet-star: Language models can teach themselves to think before speaking. *arXiv preprint arXiv:2403.09629*, 2024.

Shu Zhou, Rui Ling, Junan Chen, Xin Wang, Tao Fan, and Hao Wang. When more thinking hurts: Overthinking in llm test-time compute scaling. *arXiv preprint arXiv:2604.10739*, 2026.

Rui-Jie Zhu et al. A survey on latent reasoning. *arXiv preprint arXiv:2507.06203*, 2025.

A CIRCUIT-DEPTH THEORY: DERIVATION

We model a recurrent-depth network as resolving reasoning steps sequentially under two assumptions. (i) Each core iteration advances the latent state by at most α reasoning steps, so T iterations resolve steps up to depth $\lfloor \alpha T \rfloor$. (ii) Conditioned on sufficient depth, each of the d serial steps is executed correctly and independently with probability $p(N)$, which increases with model size. For the controlled tasks whose generator fixes serial depth, $d(x){=}K$ by construction. Under these assumptions, accuracy after T iterations is

$$\mathrm{acc}(N,T) = \mathbb{E}_{d\sim\mathcal{D}}\big[\mathbf{1}[d \le \alpha T]\, p(N)^d\big]\,. \tag{4}$$

The indicator captures depth coverage: a problem becomes solvable once the available recurrent depth αT reaches its required depth. The factor $p(N)^d$ captures the probability of executing all d serial steps correctly. The law therefore separates two effects: how far the computation reaches and how reliably it executes what it reaches.

When $p(N)$ is close to 1, the reliability term $p(N)^d = \exp(d \log p(N))$ varies slowly over the covered portion of the depth distribution. Expanding about $\bar{d} = \mathbb{E}[d \mid d \le \alpha T]$, the first-order term vanishes because the expansion is taken at the conditional mean, giving

$$\mathbb{E}\big[p(N)^d \mid d \le \alpha T\big] \approx p(N)^{\bar{d}}\Big(1 + \tfrac{1}{2}(\log p(N))^2 \,\mathrm{Var}[d \mid d \le \alpha T]\Big),$$

and therefore

$$\mathrm{acc}(N,T) \approx F_{\mathcal{D}}(\alpha T)\, p(N)^{\bar{d}}. \tag{5}$$

Equation 4 is already a reliability-weighted depth CDF, while Equation 5 gives its high-reliability factorization. As a function of $\log T$, accuracy is monotone and bounded, with its transition governed by the circuit-depth distribution and its ceiling by per-step reliability. This gives Claim 1. The theory constrains the curve to be bounded and monotone: which bounded form best describes the measured curves is determined empirically by model selection (§E).

For a fixed-depth task with $d{=}K$, the depth CDF is a step at K, so the transition occurs when $\alpha T \ge K$, giving $T^*(K) = K/\alpha$. A soft transition, such as the fitted sigmoid, shifts the half-ceiling point by an approximately constant offset, yielding $T^*(K) = K/\alpha + c$. The onset therefore remains linear in K, giving Claim 2.

Empirically, the fitted slope $1/\alpha$ decreases with model size, from $1.43{\pm}0.06$ at 19M to $0.87{\pm}0.12$ at 121M (Table 5), implying that larger models resolve each serial step with fewer recurrent iterations.

CoT advances circuit depth by generating reasoning tokens. If each step requires τ tokens, empirically 14.5 tokens per step on StateTrack (§D), reaching depth d requires approximately $L = \tau d$ generated tokens. Its accuracy is therefore governed by the same depth distribution, with the available depth growing through token generation. Sampling additional chains instead allocates compute across independent attempts, and aggregate accuracy approaches a depth-limited ceiling approximately logarithmically in the number of samples (Proposition 2).

Relaxing the independence assumption. Assumption (ii) gives the convenient form $p(N)^d$, but the depth-coverage mechanism does not require exact independence.

Correlated errors. If step successes are positively correlated, the probability of clearing all d steps can exceed $p(N)^d$. The reliability term can then be replaced by an effective depth-dependent factor that generally decays more slowly with d. This changes the ceiling and detailed reliability profile but leaves depth coverage intact, so accuracy remains monotone in T and saturates once the relevant depths are covered.

Heterogeneous resolution rate. If different steps require different numbers of recurrent iterations, replace the constant α with per-step rates α_i. The maximum resolvable depth after T iterations is

$$D(T) = \max\left\{ d : \sum_{i=1}^{d} \frac{1}{\alpha_i} \le T \right\}, \tag{6}$$

and accuracy becomes

$$\mathrm{acc}(N,T) = \mathbb{E}_{d\sim\mathcal{D}}\big[\mathbf{1}[d \le D(T)]\, p(N)^d\big]\,. \tag{7}$$

The curve is therefore still a depth CDF evaluated at the monotone effective depth $D(T)$. For a fixed-depth task,

$$T^*(K) = \sum_{i=1}^{K} \frac{1}{\alpha_i}, \tag{8}$$

which is linear in K for homogeneous per-step rates but need not be linear otherwise. Thus α and $p(N)$ should be interpreted as effective curve-level quantities. The saturating form is robust to these relaxations, whereas strict linearity of the depth law can weaken, as observed under wider-depth training (§G).

Both relaxations have empirical counterparts. The per-effective-depth ceiling fits return a prefactor above one, indicating that the first steps of a chain execute more reliably than a constant-$p(N)$ model predicts (§G). The calibrated resolution rate varies with the cost of a task's step while remaining nearly constant across the depths of a given task, since $T^*(K)$ remains linear over the measured range (§G).

Accuracy of the factored approximation. The approximation in Equation 5 incurs a correction of order $(\log p(N))^2 \operatorname{Var}[d \mid d \leq \alpha T]$ (Proposition 1). It is exact for a point-mass depth distribution and becomes tight as $p(N) \to 1$.

We evaluate the exact expectation and the factored approximation over a range of depth distributions and reliabilities (Table 2). The maximum absolute error is 0 for a point mass and remains below 0.0017 at p=0.99, even for a heavy-tailed distribution with variance 37.6. The error grows with depth variance and with $1-p$, reaching 0.24, or 64% in relative terms, only at the extreme combination of a heavy-tailed distribution and p=0.7. In that regime the factored ceiling understates exact accuracy because low-depth problems are solved more reliably than the mean-depth surrogate implies.

Our fitted regime lies near the tight end of this range. Each controlled task at fixed K has point-mass depth, while the pooled evaluation distribution has $\bar{d}\approx 3$. The measured ceilings of 0.90–0.98 imply $p \gtrsim 0.95$ under the factorized model, placing the experiments in the high-reliability regime where the approximation is accurate to a few thousandths. We therefore use the factorization to interpret the fitted ceiling in terms of per-step reliability. The depth-CDF result itself follows directly from Equation 4.

Table 2: **Error of the factored approximation.** Maximum absolute divergence from the exact expectation, $\max_T |\mathbb{E}_d[\mathbf{1}[d \leq \alpha T] p^d] - F_{\mathcal{D}}(\alpha T) p^{\bar{d}}|$, over depth distributions of increasing variance and per-step reliability p. The approximation is exact for point-mass depth and remains accurate at high p. Error increases with both depth variance and $1-p$, consistent with the predicted $(\log p)^2 \operatorname{Var}[d]$ scaling.

Depth distribution	$\operatorname{Var}[d]$	p=0.99	p=0.9	p=0.8	p=0.7
Point mass (K=3, controlled)	0.0	0.000	0.000	0.000	0.000
Poisson(3)	3.0	0.000	0.012	0.037	0.064
Geometric(0.30)	7.8	0.000	0.029	0.088	0.153
Geometric(0.15), heavy tail	37.6	0.002	0.087	0.186	0.237

B PRETRAINING RECIPE AND PERPLEXITY CERTIFICATION

All ladder models use the Huginn recurrent-depth architecture (Geiping et al., 2025): prelude l_P=1, weight-shared core l_R=4, coda l_C=1, multi-head attention with head dimension 64, SwiGLU MLP, sandwich (pre+post) normalization on both attention and MLP branches, rotary position embeddings, and Takase residual scaling ($1/\sqrt{2L}$). The core takes the concatenation of the current latent state and fixed input embedding through a linear adapter ($2d_{\text{model}} \to d_{\text{model}}$).

During training, we sample the iteration count T per step from a log-normal-Poisson distribution with mean 32, capped at twice the mean. We backpropagate through only the last 8 iterations (truncated BPTT), and earlier iterations run under `no_grad`. We use AdamW (β=0.9, 0.95), a cosine

learning-rate schedule, peak learning rate 6×10^{-4} with 400 warmup steps, weight decay 0.1, gradient clipping at 1.0, sequence length 2048, and bf16 precision. The six ladder widths are $d_{\text{model}} \in \{512, 640, 768, 896, 1024, 1280\}$, with intermediate sizes $\{1365, 1706, 2048, 2389, 2731, 3413\}$ and $\{8, 10, 12, 14, 16, 20\}$ heads.

We train on the released `tomg-group-umd/huginn-dataset`, consisting of pre-tokenized length-4097 rows with a $65{,}536$-vocabulary tokenizer. We use 2048-token windows sliced from these rows. Each model is trained on approximately 20–40 tokens per parameter, placing each near the compute-optimal training regime used for the ladder.

As a language-modeling certification, we train a separate $\sim$800M-parameter model (d_{model}=4096, two-layer core) for 0.45B tokens on identical data. It reaches validation perplexity 20.9, compared with 24.6 for a depth-matched standard transformer trained identically. The recurrent model therefore achieves lower validation perplexity than the matched standard transformer, providing evidence against a gross language-modeling deficiency as the explanation for the downstream floors and supporting their interpretation as capacity limitations on the hard tasks.

Scope of the fitted exponents. The ceiling and onset exponents ($A(N) \propto N^{0.22}$ and $\log T_0(N)$), together with the resolution rate α, are estimated under a single training-time recurrence distribution (mean 32, TBPTT window 8), held fixed across the ladder so that N is the only variable. This is a deliberate control, not a claim of schedule invariance. The training-time recurrence distribution determines the range of T on which the core is trained, so materially different schedules, such as a substantially smaller mean recurrence or a different backpropagation window, could shift T_0 or the effective α even if the saturating shape persists.

Because all model sizes use the same schedule, the reported cross-size exponents are internally consistent. Their transfer to other schedules is untested. Whether α also depends on which layers are recurrent in architectures that loop only a subset of layers remains an open question.

C CONTROLLED TASK GENERATORS

Testing a law that relates accuracy to circuit depth requires depth to be an *independent variable*. Natural benchmarks do not provide this control. We therefore construct four generators in which a task parameter determines, or tightly controls, the depth of the shortest correct solution.

Three are the *primary* domains on which the ladder is trained and the law is fit: arithmetic (StateTrack), algebraic (PermTrack), and deductive (LogicTrack). All three form linear chains of K steps. StateTrack and LogicTrack have serial depth $d(x)$=K by construction. On PermTrack, K bounds but does not fix the serial depth, making it useful for testing whether nominal and effective depth can be distinguished.

The fourth, TreeTrack, is a *stress test* that separates surface length from serial depth. Its depth is the tree height, while its visible operation count grows exponentially as 2^h-1 (§H). All four generators have multi-way answer spaces with low chance accuracy, and each admits an argument that its required depth is *irreducible*: the answer cannot be determined in fewer serial steps. Thus, above-chance performance is informative about sequential computation while the controlled depth parameter can be varied independently.

StateTrack (modular arithmetic). We sample a start digit $s \in \{0, \ldots, 9\}$ and K operations, each "add c" or "multiply by c" ($c \in \{2, \ldots, 5\}$), all modulo 10. The question gives the start and operation sequence. The target chain shows each intermediate value ("Add 4: $9 + 4 = 3$ (mod 10)") and ends "The final result is r." The answer is a single digit (10-way).

The K operations form a dependency chain: operation i+1 acts on the output of operation i, so the running value must be propagated through all K steps. Working backward is unavailable because the multipliers $\{2, 4, 5\}$ are non-invertible modulo 10, collapsing the state space and preventing recovery of the preceding state from the final value. Individual operations leave the value unchanged in 13% of sampled operations, but such an operation cannot be identified before computing the value on which it acts, so the serial depth remains K.

PermTrack (permutation composition). Five slots contain five distinct items. Each of K steps swaps a uniformly sampled pair of slots, and the task asks which slot contains a designated item (5-way). The chain shows the tracked item's slot after each swap.

Unlike the other primary tasks, K bounds rather than fixes the serial depth: the tracked item moves only when a swap touches its current slot, which occurs for 4 of the 10 slot pairs, giving an average of $0.40K$ state transitions. This is *not* the answer's serial depth, however. Whether a swap touches the tracked item depends on the state produced by earlier swaps, so every swap must still be examined. The dependency measurement below gives an effective depth of $0.995K$.

PermTrack serves two purposes. It tests whether the saturating functional form transfers to a structurally different domain and separates the cost of a reasoning step from the serial depth of the answer. The answer depends on essentially every swap, yet its onset slopes are 0.27–0.80, versus 0.87–1.43 on StateTrack at the same sizes (Table 5). The calibration in §G attributes this difference to a higher resolution rate for the cheaper permutation step.

LogicTrack (symbolic deduction). We sample a path $c_0 \to c_1 \to \cdots \to c_K$ through distinct categories drawn from 10 labels, expressed as K rules "every c_i is a c_{i+1}", and add up to K distractor rules originating off the path. We never add a second out-edge from a path category, so the chain remains deterministic. The rules are shuffled, the start is c_0, and the answer is the terminal category c_K (10-way).

Because distractors are interleaved, neither rule order nor a local guess identifies the answer. Each of the K hops requires selecting the applicable rule and depends on the previous hop, giving serial depth K.

TreeTrack (branching evaluation). A balanced binary expression tree of height h over mod-10 addition/multiplication is evaluated recursively: leaves are digits, internal nodes combine children, and the answer is the root value. The visible problem contains 2^h-1 operations, but the irreducible serial depth is the tree height h, since each level depends on the one below. Thus circuit depth is

$$h = \log_2(\#\text{ops}+1),$$

separating circuit depth from operation count by an exponential factor. We train on $h \in \{1, \ldots, 4\}$.

Training and evaluation protocol. Each generator exposes depth as a parameter, and we fine-tune every size and seed on two ranges. The narrow range is $K \in \{1, \ldots, 5\}$ for all four tasks, where the onset $T^*(K)$ is measurable. The wide range is the largest each generator admits: $K \leq 12$ for StateTrack and $K \leq 9$ for LogicTrack, where the accuracy ceiling still varies with N (§F). Fine-tuning uses 30k examples at seed 42 and evaluation uses held-out problems at seed 999. A three-stage curriculum progressively replaces the visible chain with direct-answer supervision for latent models, while CoT baselines receive the full generated chain.

Effective serial depth on StateTrack. Nominal depth K need not equal the serial depth an instance requires. Because multiplication by an even number or by 5 is non-injective on $\mathbb{Z}_{10}$, a suffix of operations can compose to a constant map, after which the answer no longer depends on the starting value or earlier operations. We therefore define effective depth as the smallest s such that the composition of the last s operations is constant on $\mathbb{Z}_{10}$, and as K when no such suffix exists. The definition is well posed because constancy is monotone in s: prepending an operation to a constant map leaves it constant.

Across 1000 instances per depth, the mean effective depth is 4.55 at K=5 ($0.91K$), 6.25 at K=8 ($0.78K$), and 7.71 at K=12 ($0.64K$), and the fraction with effective depth below K rises from 0.26 to 0.73. Two checks rule out simpler explanations. The running value reaches its final value only at step $0.95K$–$0.98K$, so the effect is loss of dependence on early operations rather than the state simply freezing. The answer distribution has modal probability 0.17–0.27, so a constant-answer heuristic scores near 0.25, well below the observed 0.6 plateaus. For $K \leq 5$, nominal and effective depth agree within 10%, so we index by K. Beyond this range, we stratify by effective depth as in §G.

Measuring serial depth from task labels alone. The algebraic construction above uses the task's operation semantics, which may not be available for a new task. For a chain-structured task in which one state is updated at each step, the same dependency depth can be recovered from labels alone.

Resample step j from the task's step distribution, recompute the gold answer with every other step fixed, and call step j *live* if the answer changes. The live steps form the suffix that survives the last information-destroying step, so their count estimates the required serial depth without a trained model.

The two measures agree closely. On StateTrack, the label-only count matches the algebraic suffix length on $99/98/96/95\%$ of instances at K=3/5/8/12, with mean depths $2.96/4.45/6.34/7.72$ versus $2.97/4.47/6.38/7.78$. Across the three tasks, the resulting mean dependency depth per nominal unit is ρ=0.947 on StateTrack at $K \leq 5$, 0.764 at $K \leq 12$, 1.000 on LogicTrack at $K \leq 5$, 0.983 at $K \leq 9$, and 0.995 on PermTrack. LogicTrack never collapses because each rule is a bijection on categories. PermTrack likewise does not collapse, so its $0.40K$ state-transition count should not be confused with serial depth.

Instance overlap between training and evaluation. Because both splits are sampled from the same generator under different seeds, overlap is unavoidable at shallow depths where the instance space is small: StateTrack has only 72 distinct instances at K=1. We report this overlap rather than filtering it, which would bias evaluation toward instances missed by the training sampler. In a 200-problem StateTrack evaluation set, overlap falls from 100% at K=1–2 to 68%, 14%, and 4% at K=3, 4, 5, and is 0% for all $K \geq 6$. PermTrack likewise overlaps fully at $K \leq 2$ and by at most 4% for $K \geq 4$, while LogicTrack overlaps only at K=1 (35%). Thus the depths driving the depth law and wide-depth results are effectively disjoint from training, and the shallow overlapping depths are precisely those at which every size already saturates.

Verbatim examples. Table 3 gives one generated instance per task at depth K=3 (h=2 for TreeTrack), exactly as presented to the model: prompt, step-by-step target, and answer. Latent models receive answer-only supervision in the final curriculum stage. At evaluation, the chain shown in the table must be resolved entirely inside the recurrence.

Table 3: **One verbatim instance per controlled task** at depth K=3 (TreeTrack h=2, whose 3 operations span 2 serial levels). Prompts and targets are reproduced as generated, and the answer is a single label (10-way, or 5-way for PermTrack).

StateTrack	**Prompt:** "Start with the number 2. Then multiply by 2, then add 2, then multiply by 2. All arithmetic is modulo 10. What is the final result?" **Target chain:** "Start with 2. Multiply by 2: $2 * 2 = 4$ (mod 10). Add 2: $4 + 2 = 6$ (mod 10). Multiply by 2: $6 * 2 = 2$ (mod 10). The final result is 2." **Answer:** 2
PermTrack	**Prompt:** "Five items sit in slots 0 to 4: slot 0 has red, slot 1 has blue, slot 2 has green, slot 3 has gold, slot 4 has black. Then swap slots 1 and 3, then swap slots 0 and 4, then swap slots 4 and 0. Which slot holds 'red' at the end?" **Target chain:** "... Track where 'red' is. Swap slots 1 and 3. Now 'red' is in slot 0. Swap slots 0 and 4. Now 'red' is in slot 4. Swap slots 4 and 0. Now 'red' is in slot 0. The final slot is 0." **Answer:** 0
LogicTrack	**Prompt:** "Rules: every 0 is a 7; every 7 is a 0; every 9 is a 8; every 2 is a 6; every 6 is a 0; every 1 is a 6. Start: 2. Follow the rules step by step for 3 step(s). What is the final category?" **Target chain:** "Start with 2. Step 1: every 2 is a 6, so now 6. Step 2: every 6 is a 0, so now 0. Step 3: every 0 is a 7, so now 7. The final category is 7." **Answer:** 7
TreeTrack	**Prompt:** "Evaluate this expression modulo 10, resolving inner parentheses first: $((0 * 1) + (5 + 9))$ = ?" **Target chain:** "$0 * 1 = 0$ (mod 10). $5 + 9 = 4$ (mod 10). $0 + 4 = 4$ (mod 10). The final result is 4." **Answer:** 4

D COMMENSURABLE FLOP ACCOUNTING

We count $2PS$ FLOPs for a forward pass of P non-embedding parameters over S tokens.

Latent (recurrent-depth): the prelude and coda run once over the context, while the core runs T times:

$$\mathrm{FLOPs}(T) = 2(P_{\text{pre}}+P_{\text{coda}})S_{\text{ctx}} + T \cdot 2(P_{\text{core}}+P_{\text{adapter}})S_{\text{ctx}} + \text{decode}.$$

The per-iteration cost is therefore the core plus adapter, a fraction $l_R/(l_P+l_R+l_C)=4/6$ of a full layer stack, evaluated on the short context.

CoT: generating L reasoning tokens autoregressively costs $2PS_{\text{prompt}} + L \cdot 2P$, corresponding to one full forward per generated token.

Coconut: each continuous thought is a full single-position forward, costing $c \cdot 2P$ in addition to the prompt.

These conventions place recurrent depth and token generation on a common FLOP scale (Figures 8, 6).

The token budget in units of reasoning steps, and what transfers. The FLOP accounting prices iterations and generated tokens. Converting either into reasoning steps additionally requires the task's verbalization cost. Regressing median gold-chain length on depth gives $14.5K+15.7$ tokens on StateTrack, $13.0K+16.0$ on LogicTrack, and $14.0K+42.0$ on PermTrack, each with $R^2=1.0000$. The intercept is fixed preamble, largest on PermTrack because its chain restates all five slot contents.

The linear relation also holds on GSM8K: across all 1319 test problems, restricting to strata with at least 40 examples and defining depth by newline-separated reasoning lines gives 29.5 tokens per line at $R^2=0.997$, with a 10th–90th percentile range of 23–43 tokens. Thus the token cost of a reasoning step depends on task and tokenizer, whereas linearity is the transferable property. In our setting one recurrence costs roughly 32 tokens because 4 of 6 layers form the core and the prompt is 48 tokens long. More generally that ratio scales with the core-to-stack depth ratio and with prompt length, since a recurrence reprocesses the context while an autoregressive token is incremental against a KV cache. We therefore report task-specific crossover budgets rather than a universal token equivalent.

Why the token curve is a chain-length CDF. A chain earns credit only after termination. Hence, at budget L, a problem is solved exactly when its gold chain fits within L tokens and the model reproduces it. Because the CoT baselines reproduce the gold chain whenever the budget admits it, accuracy equals the fraction of held-out problems whose chain length is at most L. Predicted and measured accuracy agree to three decimals at all 11 budgets on both StateTrack and PermTrack, with maximum absolute discrepancy 0.000. Consequently the curve is identical across model sizes to three decimals, while its crossover with latent depth depends on verbalization cost: PermTrack's longer preamble shifts the crossover from 4.1×10^9 to 6.3×10^9 FLOPs. Scoring unterminated chains by trailing digits violates this identity and inflates low-budget accuracy by up to 0.33, so we require termination.

Latency and memory. The accounting above counts FLOPs, which do not capture two serving-relevant properties. Recurrent iterations execute serially, so latency grows linearly in T unless loop execution is parallelized across tokens, as in recent architectural work (Wu et al., 2025). Token generation is also serial in L but extends the sequence, growing the KV cache with each generated or continuous-thought position, whereas recurrence reprocesses a fixed set of positions. The frontiers of Figures 8 and 7 are therefore budget statements, and a deployment weighing latency or memory may prefer a different operating point on the same law.

E Full model-selection table

We fit a sigmoid $A/(1+e^{-k(\log x - x_0)})$, exponential saturation $A(1-e^{-x/\tau})$, power law $A-Bx^{-\alpha}$, and log-linear form $a+b\log x$ to each clean latent curve and rank them by AIC. The 59M model, added to densify the N-axis, has latent sweeps only and therefore contributes to joint fitting and model selection but not to axis comparisons.

The ladder contributes 54 latent curves, one per task, size, and seed ($3\times6\times3$). A saturating form is AIC-preferred on every curve: sigmoid on 30, exponential saturation on 22, and power law on 2. Log-linear is never preferred. By R^2, the best saturating form exceeds log-linear by 0.019–0.303 (median 0.137) on all 54 curves. The power-law fits are themselves concave and bounded over the

Table 4: **Model selection on all 54 latent curves.** Per-curve R^2 for the four candidate forms of §4, as ranges over model sizes and seeds. No curve prefers the unbounded form: the best saturating fit leads log-linear by 0.019–0.303 (median 0.137). The AIC winners are sigmoid 30, exponential-saturation 22, power-law 2 (§E).

		Saturating forms			Unbounded
Model	**Task (domain)**	Sigmoid	Exp-sat	Power-law	Log-linear
Huginn-3.5B	GSM8K (math)	0.997	0.878	0.762	0.762
Ladder, 19–121M	StateTrack (arithmetic)	0.965–0.997	0.915–0.993	0.948–0.998	0.856–0.965
Ladder, 19–121M	PermTrack (permutation)	0.927–0.999	0.927–0.999	0.877–0.994	0.696–0.885
Ladder, 19–121M	LogicTrack (deduction)	0.944–1.000	0.953–0.999	0.894–0.993	0.738–0.881

observed range, so the model-selection result is specifically against an unbounded log-linear rise. Removing the largest-T point leaves the same result. The released Huginn GSM8K curve and the off-ladder 174M StateTrack curve of §G are also sigmoid-preferred and are tallied separately, so the 54 count exactly matches the ladder design.

Saturation alone could reflect a generic capability ceiling. That explanation predicts neither a depth-ordered onset nor sensitivity to tree height when surface operation count grows exponentially. Both effects are observed (§5.2), supporting depth coverage as the mechanism behind the saturation.

On the Huginn GSM8K curve, the fitted power-law exponent approaches zero, making $A-Bx^{-\alpha}$ numerically indistinguishable from $a+b\log x$ and yielding the same R^2=0.762 in Table 4. Per-curve R^2 and AIC values are included in the released sweep outputs.

F Compute-optimal training and the N-axis confound

An earlier ladder trained every size on the same absolute token budget ($\approx$ 0.5B), producing an inverted N-axis: a 150M model outperformed an 800M model on StateTrack because the fixed budget was near-optimal for the smaller model but severely undertrained the larger one. Scaling the token budget with model size to approximately 20–40 tokens per parameter removes this confound and restores the expected rising ceiling $A(N) \propto N^{0.22}$ (§5.1). We therefore use size-matched token budgets throughout the ladder.

Seed robustness of the ceiling. Saturation ceilings averaged over three fine-tuning seeds are 0.77±0.05 (19M), 0.92±0.03 (30M), 0.98±0.03 (44M), 1.00±0.00 (59M), 0.96±0.03 (78M), and 0.99±0.01 (121M). The narrow $K \leq 5$ range therefore saturates from 44M upward, leaving little information about the N-dependence of the ceiling. The corresponding fit gives $A(N) \propto N^{0.10}$ with cluster-bootstrap 95% CI $[0.00, 0.17]$.

We therefore treat the narrow-ladder exponent as weakly identified rather than as a point-wise monotonic ranking of model sizes. Fine-tuning uses three seeds (42/202/303) in every latent, sequence, and wide-depth cell. Each reported curve is a three-run mean. The pretrained backbones are single-run per size because pretraining is shared across all fine-tuning cells, as are the $K \leq 5$ CoT baselines because their curves are determined by the chain-length distribution (§D). The $K \leq 12$ CoT baselines use three seeds.

The wide-depth ladder, and why the ceiling exponent needs it. Fine-tuning on $K \in \{1, \ldots, 5\}$ leaves StateTrack solved from 44M upward, pinning the ceiling near one. We therefore retrain every size and seed on $K \in \{1, \ldots, 12\}$ with the same pretrained backbones, optimizer, schedule, and seeds. The resulting ceilings are 0.67±0.01, 0.76±0.03, 0.95±0.06, 1.00±0.00, 0.91±0.09, and 1.00±0.00 from 19M to 121M. The joint surface fits R^2=0.948 on 108 measurements, and a cluster bootstrap over whole curves gives a_A=0.217 with 95% CI $[0.11, 0.26]$. The wide-depth training loss also remains near 0.2–0.4, indicating that these evaluations remain capacity-limited rather than ceiling-limited.

LogicTrack has maximum depth 9, since its generator samples K+1 distinct categories from ten labels. Its narrower range gives ceilings from 0.78 to 0.89 and a_A=0.07 with CI $[0.00, 0.09]$. We

therefore use wide-depth StateTrack to identify the ceiling exponent and treat LogicTrack as a directional cross-domain check.

A bounded ceiling variant. Equation 3 uses a local power-law ceiling, which is unbounded when extrapolated. At 121M and 3.5B, the wide-depth fit reaches 1.19 and 2.47, respectively. To test whether this parameterization drives our conclusions, we refit both ladders with an equal-parameter logistic ceiling in $\log N$, $A(N) = 1/(1 + (n_0/N)^\gamma)$.

On the $K \leq 5$ ladder, the bounded form fits the joint surface comparably (R^2=0.964 vs. 0.962) and predicts the held-out 121M curve slightly better (R^2=0.946, MAE 0.044 vs. R^2=0.930, MAE 0.045). At T=16, 32, it predicts 0.93, 0.97, compared with 1.05, 1.11 from the power law. On the wide-depth ladder, the power law fits slightly better in sample (R^2=0.948 vs. 0.933), while the allocation exponents change from $N^* \propto C^{0.42}$ and $T^* \propto C^{0.69}$ to $C^{0.33}$ and $C^{0.79}$.

Thus the qualitative allocation policy is unchanged: the optimum begins at the smallest model and shifts toward larger models as compute grows, while a bounded ceiling allocates more of the additional budget to depth after saturation. The exponent values vary by roughly 0.1 across parameterizations, so we interpret them as properties of the fitted ladder surface rather than universal constants.

G Out-of-distribution depth generalization

A model fine-tuned only on $K \leq 5$ could learn depth-specific routines rather than a general resolution process. We therefore evaluate all six ladder sizes on unseen depths $K \in \{6, \ldots, 9\}$, using 200 instances per depth and iteration count (per-cell sampling noise ± 0.035). Latent accuracy increases with T at every size, from 0.24–0.31 at T=1 to 0.59 (19M), 0.65 (30M), 0.77 (44M), 0.96 (59M), 0.84 (78M), and 0.88 (121M) at T=32. Larger models therefore resolve more unseen depths, while preserving the in-distribution ceiling ordering, including the 59M–78M inversion of §F. This is consistent with the size dependence of the reliability term $p(N)$.

Accuracy also decreases with depth within the unseen range, as predicted by the theory. At T=32, per-K accuracies at 44M are 0.84/0.80/0.77/0.69 for K=6, ..., 9, and at 121M are 0.93/0.92/0.89/0.79. The other four sizes show the same trend up to one K=6–7 tie or inversion of at most 0.04. Per-K onsets are noisier out of distribution: they are non-decreasing in K at 44/59/78/121M, but invert once at 19M and 30M. We therefore treat linearity of $T^*(K)$ as an in-distribution result, while the weaker prediction that additional recurrence improves performance on deeper unseen problems holds beyond the training range. An earlier token-confounded ladder was flat on these same depths, indicating that this extrapolation depends on the compute-optimal training used here.

Nominal depth can fail as a proxy for serial depth. The stronger test is whether the onset continues to scale with the generator's depth parameter when the training range is widened. We retrain the 44M and 78M models on StateTrack with K uniform on $\{1, \ldots, 9\}$, using three seeds and 200 instances per depth and iteration count. For $K \leq 4$, the onsets rise with depth: T^*=1, 2, 3, 4 at 44M, giving slope 1.00, and 2, 3, 4, 4 at 78M, giving slope 0.70. For $K \geq 5$, the onsets compress into a few grid points: 4, 4, 5, 4, 5 at 44M and 5, 5, 7, 5, 5 at 78M for K=5, ..., 9. Thus K=8 turns on before K=7 at both sizes. Fitting the full range gives slopes of only 0.43 (R^2=0.79) and 0.42 (R^2=0.64), respectively. Yet depth continues to separate accuracy: at T=32, accuracy falls from 1.00 at K=1 to 0.75 at K=9 at 44M, and from 1.00 to 0.68 at 78M.

This breakdown occurs because nominal K ceases to measure the computation required by the answer. StateTrack composes additions and multiplications modulo 10: multiplication by an even number or by 5 is non-injective, so a suffix of operations can compose to a constant map, after which the answer no longer depends on the starting value or any earlier operation. The smallest such suffix gives a mean effective depth of $0.91K$ at K=5 and $0.64K$ at K=12, with 73% of K=12 instances requiring fewer than K serial steps. The loss of onset ordering therefore coincides with the divergence between nominal and effective depth.

Re-indexing by effective depth restores the law. When instances are re-indexed by effective depth, the onset again increases with the computation required by the answer. Across the six ladder sizes, 547 depth pairs are concordant and 17 discordant (Kendall τ=0.85, p<10^{-100}), with no discordant pairs at the four largest sizes. The linear slope is 0.47–1.98 with R^2=0.67–0.84. Thus

the onset law tracks serial depth rather than the generator's nominal depth parameter (§5.2). The earlier flattening reflects a failure of nominal depth as a proxy, not a failure of the depth law. The saturating-CDF form likewise persists across effective depths.

The depth ordering, tested by rank. The slope of $T^*(K)$ depends on the resolution of the T grid. We therefore first test the ordering predicted by the theory: deeper problems should turn on later. Across the 54 curves, comparing every pair of depths $K_i{<}K_j$ gives 461 concordant pairs, 4 discordant pairs, and 63 ties at grid resolution (Kendall $\tau{=}0.87$, sign test $p{<}10^{-100}$). The corresponding concordant/discordant counts are $168/1$ on StateTrack, $160/0$ on LogicTrack, and $133/3$ on PermTrack. The Spearman correlation between K and T^* is positive on all 54 curves. The depth ordering therefore does not depend on a linear fit.

Full slope table and resolution-rate confidence intervals. Table 5 reports the slope of $T^*(K)$ for each size and domain, averaged over three fine-tuning seeds. Onsets are measured on a 13-point T grid with unit spacing through $T{=}8$, using 200 instances per depth and iteration count. StateTrack slopes range from $1.43{\pm}0.06$ at 19M to $0.87{\pm}0.12$ at 121M, while LogicTrack follows the same trend from $1.60{\pm}0.30$ to $0.87{\pm}0.12$. Every per-seed fit is well determined ($R^2{=}0.81$–1.00). Since the slope is $1/\alpha$, the decrease with model size corresponds to a higher depth resolution rate.

PermTrack has consistently smaller slopes, 0.27–0.80. This does not reflect a shorter dependency chain: the answer depends on $0.995K$ of its swaps (§C). Instead, PermTrack steps are resolved at a higher rate. The calibration below estimates nearly two steps per iteration, compared with approximately one for StateTrack and LogicTrack, consistent with a swap requiring less computation than a modular multiply.

Table 5: **The depth-law slope is ≈ 1 recurrence per circuit-depth level, across three reasoning domains and six model scales.** Slope of $T^*(K)$ per size, mean$\pm$std over three fine-tuning seeds. All 54 per-seed fits are well determined ($R^2{\geq}0.70$, ≥ 3 usable onsets). The slope is near one in both an arithmetic and a deductive domain, the theory's central prediction. PermTrack sits near 0.5, and §G attributes that to a higher resolution rate for a cheaper step, since its answer depends on $0.995K$ of its swaps.

Domain	19M	30M	44M	59M	78M	121M
StateTrack (arithmetic)	$1.43{\pm}0.06$	$1.30{\pm}0.10$	$0.90{\pm}0.10$	$0.93{\pm}0.15$	$1.07{\pm}0.15$	$0.87{\pm}0.12$
LogicTrack (deduction)	$1.60{\pm}0.30$	$1.23{\pm}0.21$	$1.17{\pm}0.21$	$1.10{\pm}0.17$	$1.03{\pm}0.15$	$0.87{\pm}0.12$
PermTrack (permutation)	$0.80{\pm}0.17$	$0.63{\pm}0.12$	$0.47{\pm}0.06$	$0.63{\pm}0.12$	$0.27{\pm}0.21$	$0.40{\pm}0.17$

The corresponding resolution rate $\alpha{=}1/\text{slope}$ on StateTrack, with bootstrap 95% CIs over the per-K onsets, is 0.70 $[0.64, 0.87]$ (19M), 0.77 $[0.64, 0.90]$ (30M), 1.11 $[0.84, 2.00]$ (44M), 1.07 $[0.86, 1.50]$ (59M), 0.94 $[0.72, 1.50]$ (78M), and 1.15 $[1.00, 1.60]$ (121M). Every interval excludes zero and lies within a factor of two of one, and α increases with size. The intervals are based on five onsets per fit and therefore provide coarse rate estimates. In contrast, the continuous-thought axis shows no rising per-K onset, so its α is not identifiable.

Per-step reliability measured from per-effective-depth ceilings. Equation 3 identifies the ceiling with $p(N)^{\bar{d}}$ under the factorized approximation. At a fixed depth d, the sharper prediction is $A_d(N){=}p(N)^d$, so $\log A_d$ should be linear in d with slope $\log p(N)$. We test this prediction using the wide-depth models, stratifying instances by effective depth $d \in \{1, \ldots, 10\}$ at 100 instances per depth and iteration count and taking A_d as the plateau of each depth-specific curve.

The prediction holds at depths with headroom and is measurable at four of the six sizes. At 59M and 121M, every effective depth up to 10 is solved outright, so these sizes provide no information about p and are excluded. The remaining fits give $p{=}0.823$ (19M), 0.844 (30M), 0.964 (44M), and 0.948 (78M), with $R^2{=}0.95/0.94/0.86/0.83$. The recovered reliabilities scale as $p(N) \propto N^{0.118}$. The shallowest depths are often saturated at 1.00 and are therefore excluded from the geometric fit, since including them flattens the regression. The fitted prefactor exceeds one, at 1.2–2.4, indicating that the earliest steps are executed more reliably than later ones. Thus $p(N)$ should be viewed as an effective later-step reliability rather than a literal constant across all steps.

The same analysis on a released model provides a cross-scale check. Across all 1319 GSM8K test problems, Huginn-3.5B's per-stratum plateau falls from 0.571 at two gold reasoning lines to 0.150 at seven, with $\log A_d$ linear in depth at R^2=0.961 and p=0.747 (§K). The reliability component of the law is therefore visible both on the controlled ladder and on a natural benchmark at $29\times$ the largest ladder size.

Calibrating the depth rule for a new task. Claim 2 predicts $T^*=d/\alpha$ in serial depth. For a task whose nominal parameter is only a proxy for serial depth, we estimate the conversion factor $b = \mathrm{d}d/\mathrm{d}K$ from task labels. The resulting values are 0.875 on StateTrack at $K \leq 5$, 0.582 at $K \leq 12$, 0.999 on LogicTrack at $K \leq 5$, 0.955 at $K \leq 9$, and 0.996 on PermTrack. Correcting the onset slope gives $\alpha=b/\text{slope}$, reported in Table 6.

Table 6: **Correcting nominal depth to dependency depth gives the two chain-arithmetic tasks a single resolution rate per model size.** $\alpha=b/\text{slope}$, with b measured from task labels alone and the slope from Table 5. StateTrack and LogicTrack agree to within 0.15 at five of six sizes and rise together with model size. PermTrack does not join them, and its dependency depth is $0.995K$, so its rate is genuinely about twice as large.

Task	19M	30M	44M	59M	78M	121M
StateTrack (arithmetic)	0.61	0.67	0.97	0.94	0.82	1.01
LogicTrack (deduction)	0.62	0.81	0.86	0.91	0.97	1.15
PermTrack (permutation)	1.25	1.57	2.13	1.57	3.74	2.49

The correction aligns the two tasks with comparable step costs: their rates agree to within 0.15 at five of six sizes and increase from about 0.6 at 19M to about 1.0 at 121M. It does not make α universal. PermTrack resolves about two steps per iteration, so its larger rate reflects the lower cost of a swap rather than a model-only property.

The same calibration is largely consistent across depth ranges. The wide-depth StateTrack ladder gives α=1.16 (44M), 0.93 (59M), 1.79 (78M), and 1.33 (121M), compared with 0.97, 0.94, 0.82, and 1.01 on the narrow ladder. Three of four agree within 0.35, and the 78M discrepancy coincides with the poorest onset fit (R^2=0.50). LogicTrack at $K \leq 9$ is not informative because its onsets compress to a slope of 0.10 at $R^2 \leq 0.11$. Thus, when onset curves remain separated, task labels provide a direct way to estimate serial depth and calibrate α for allocating recurrent compute. We do not claim that α transfers across tasks with different step costs.

Reading theoretical quantities from the fitted parameters. The saturating fit can be estimated without assuming the mechanism in Equation 2. We therefore distinguish directly measured quantities from those inferred through the factorized approximation.

Resolution rate α: estimated directly as $\alpha=1/\text{slope}$ from the per-K onset curves, giving α=0.62–1.25 across sizes.

Per-step reliability $p(N)$: under the factorized model, $A(N)=p(N)^{\bar{d}}$. For the controlled-task curves, $\bar{d}\approx 3$ when $K \sim \mathrm{Unif}\{1,\ldots,5\}$. The ceilings $0.77/0.92/0.98/1.00/0.96/0.99$ at $19/30/44/59/78/121$M imply $p(N)$=$0.92/0.97/0.99/1.00/0.99/1.00$, scaling approximately as $N^{0.041}$. Equivalently, dividing the fitted ceiling exponent 0.122 by $\bar{d}$ gives the same 0.041. This is an interpretation of the fitted ceiling, not an independently identified quantity. At the largest size, $p(N)$ reaches the boundary and is therefore only a lower bound.

Depth distribution $\mathcal{D}$: for a fixed-depth task, $F_{\mathcal{D}}$ is a step and its 0.5 crossing identifies K/α. For mixed-depth tasks, normalized per-iteration accuracy increments estimate the probability mass in each covered-depth bin, as in the inversion of Proposition 1 reported for SVAMP and GSM8K in §K. The mean and spread of $\mathcal{D}$ then follow up to the unknown α scale.

The sigmoid steepness k is not a separate physical constant. It reflects the width of $F_{\mathcal{D}}$ in $\log T$: it is sharper for a point-mass depth distribution and broader for a distributed one, explaining the larger fitted k on controlled tasks than on natural benchmarks. Without external depth annotation, α and the depth scale are not separately identifiable from a single curve. We therefore estimate α from onset slopes and $p(N)$ from per-depth ceilings.

No overthinking within the tested range. Additional test-time compute can degrade accuracy: longer sampled chains can abandon correct answers (Zhou et al., 2026), and looped models can degrade when iterated beyond the depth required by a task. We observe no such decline within the tested range. Across the full ladder on StateTrack and all three Huginn real benchmarks, accuracy shows no decline with additional iterations through the largest tested T: the peak-to-final drop is at most 0.028 (mean 0.002 across the 54 ladder curves), within per-point sampling noise (± 0.07 at n=50, ± 0.035 at n=200).

Equation 4 predicts a plateau rather than a decline: once αT exceeds a problem's depth d, further iterations do not increase the covered depth. A decline would require additional iterations to corrupt an already resolved state, which is outside the present model. Our sweeps reach $T \leq 32$, roughly $6\times$ the maximum training depth. We therefore make no claim about much larger budgets or training schedules that do not teach a stable fixed point.

Shape persistence at 174M. We fine-tune an additional pretrained model (174M non-embedding, $1.4\times$ the largest ladder size) on TreeTrack and sweep $\mathrm{acc}(T)$. The curve remains saturating: sigmoid R^2=0.94 versus log-linear 0.78, with a plateau near 0.59 that leaves substantial headroom. The same model on StateTrack is also sigmoid-preferred (R^2=0.97 versus log-linear 0.84), reaching 1.00 and therefore saturating by ceiling effect.

We use this experiment only as a shape-persistence check. TreeTrack's ceiling is nearly size-independent ($A(N) \propto N^{0.03}$, §H), so it cannot test the ceiling law $A(N) \propto N^{0.22}$. A joint-law fit on the 19–121M TreeTrack ladder predicts the 174M curve no better than directly reusing the 121M curve (both MAE $\approx$0.02). Quantitative ceiling-law extrapolation at this scale therefore requires a task with both headroom and a size-dependent ceiling, which none of our controlled tasks provides. The quantitative extrapolation claim remains the $1.6\times$ StateTrack test in §5.1, with the released 3.5B Huginn providing the structural cross-scale anchor.

H Additional tasks (PermTrack, LogicTrack, TreeTrack): full results

This appendix gives the full results for the additional task domains used to test domain generality, the distinction between circuit depth and operation count, and task dependence of the compute-optimal allocation law (§5.1, §5.2, §5.4).

PermTrack. PermTrack tests whether the saturating dependence on recurrent depth extends beyond arithmetic reasoning. Across all six model sizes, accuracy increases with T and then saturates, rising from 0.28–0.52 at T=1 to 0.96–1.00 by T=8, averaged over three fine-tuning seeds. The curve therefore has the same qualitative form as StateTrack despite having a different computation structure. It saturates at smaller T, however, and its depth-law slope is correspondingly smaller. As shown in §G, this difference is explained by a higher resolution rate for the cheaper permutation step rather than by a shorter dependency chain.

PermTrack is also within the capacity of all six models: its fitted ceiling scales as $A(N) \propto N^{0.005}$. The resulting compute-optimal allocation therefore places almost all additional compute on depth,

$$N^*(C) \propto C^{0.07}, \qquad T^*(C) \propto C^{1.06},$$

in contrast to $N^*(C) \propto C^{0.42}$ on the wide-depth StateTrack ladder. Thus, the allocation law depends on how strongly model scale limits the task ceiling.

We also compare recurrent depth with CoT at matched FLOPs (Figure 7). The latent-depth frontier remains ahead to a larger budget on PermTrack, with CoT overtaking at 6.3×10^9 FLOPs compared with 4.1×10^9 on StateTrack. The difference follows from the chain-length distribution: a PermTrack chain contains a 42-token preamble listing the five slot contents, compared with 16 tokens on StateTrack. The token axis therefore pays a larger fixed verbalization cost before reaching the same circuit depth. The relative efficiency of the two axes is consequently task dependent rather than an intrinsic property of either computation mechanism.

LogicTrack. LogicTrack provides a third reasoning domain in which the model must compose a sequence of symbolic deductions. Its $\mathrm{acc}(N, T)$ surface again has the same saturating form. The

joint fit over six sizes and three seeds gives R^2=0.94 over 108 measurements, with a ceiling scaling as $A(N) \propto N^{0.16}$. The 19M model reaches a plateau of 0.66, while the 121M model reaches 0.93.

The fitted ceiling exponents differ across tasks: $N^{0.005}$ for PermTrack, $N^{0.12}$ for StateTrack, $N^{0.16}$ for LogicTrack, and $N^{0.03}$ for TreeTrack. This variation is consistent with the interpretation in Equation 2: when a task is already within the capacity of the smallest models, increasing N has little effect on its ceiling, whereas capacity-limited tasks retain a stronger dependence on model size. StateTrack is therefore the primary axis for estimating the N-dependence, while PermTrack and LogicTrack test whether the functional form persists across domains.

TreeTrack: separating circuit depth from operation count. TreeTrack provides a direct test of whether test-time compute tracks serial depth rather than the number of visible operations. The input is a balanced binary arithmetic-expression tree evaluated modulo 10. Its irreducible serial depth is the tree height h, while the number of visible operations is 2^h-1 (§C).

The saturating dependence on T persists across all six ladder sizes. The sigmoid fit gives R^2=0.975/0.987/0.998/0.995/0.993/0.998 at 19/30/44/59/78/121M, compared with R^2=0.72–0.90 for the log-linear form. The ceiling remains between 0.49 and 0.56 and is nearly independent of model size (§G), since the tested heights $h \leq 4$ are within the capacity of all ladder models.

The dependence on depth follows tree height (Figure 5): height-1 trees saturate by T=4, while deeper trees require more iterations and reach lower ceilings. This ordering is inconsistent with a simple operation-count hypothesis, under which the required compute would grow as 2^h-1. Because the deepest trees also encounter a substantial accuracy ceiling, we use TreeTrack primarily to test the ordering and saturating form rather than fitting a per-height onset slope.

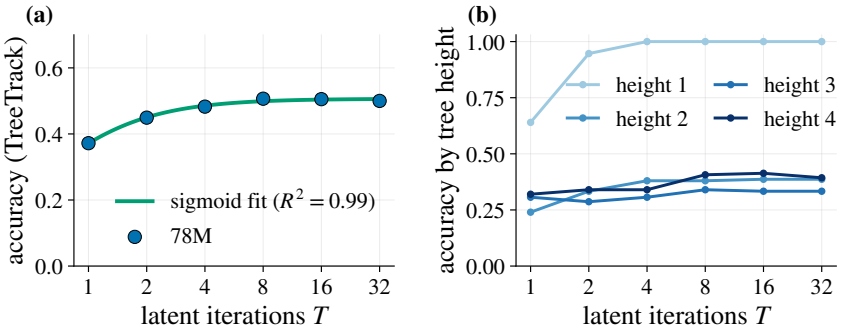


Figure 5: **The saturating law holds on a branching task (TreeTrack) where circuit depth is the tree height, not the step count.** **(a)** Overall accuracy vs latent iterations T for the 78M model, sigmoid R^2=0.99. **(b)** Accuracy by tree height: shallow (height-1) trees saturate to 1.0 quickly, deeper trees rise more slowly and plateau lower, as the circuit-depth reading predicts.

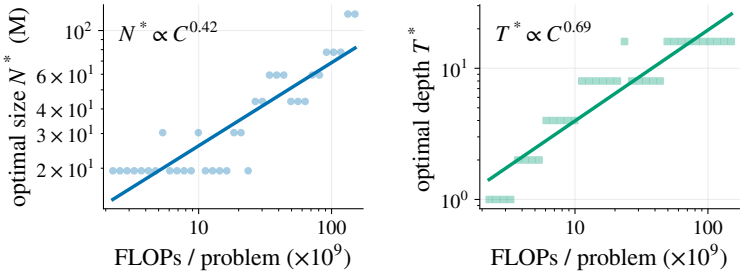


Figure 6: **Compute-optimal allocation of an inference budget (the law of §5.4).** As the per-problem FLOP budget grows, the optimal split scales model size as $N^*(C) \propto C^{0.42}$ and latent depth as $T^*(C) \propto C^{0.69}$ on the wide-depth StateTrack ladder, with the optimum moving from 19M at one iteration to 78M at sixteen. The exponents are task-dependent: on PermTrack, whose ceiling is size-insensitive, the optimum spends almost entirely on depth ($N^* \propto C^{0.07}$).

I THE SEQUENCE AXIS (CONTINUOUS THOUGHTS): MECHANISM, TRAINING, AND FULL RESULTS

Matched comparison. We next ask whether adding latent sequence positions provides the same scaling as increasing recurrent depth (§5.2). We evaluate a Coconut-style continuous-thought model on the same controlled tasks using a standard-transformer twin matched to each recurrent model in scale and pretraining budget.

A continuous thought feeds a hidden state back as the next input embedding, requiring a full forward pass. The recurrent-depth model instead repeatedly applies its weight-shared core. Because these computations cannot share the same architecture, each recurrent model is paired with a standard-transformer twin having the same hidden width, head configuration, feed-forward width, and total layer count, pretrained on the same corpus. The pretraining token budgets differ by at most 4% (0.48 versus 0.50B at 19M and equal to two significant figures at 30, 78, and 121M). Five of the six ladder sizes have such twins, excluding 59M (§E).

One residual difference remains: the depth axis uses a core pretrained to be iterated, while the sequence axis uses a conventionally pretrained twin. A within-twin CoT control removes it. CoT baselines are fine-tuned from the same standard-transformer weights and on the same tasks. With budgets from 8 to 128 tokens, their accuracy increases from 0.09 to 0.83 at 19M, 0.10 to 0.98 at 30M, 0.09 to 0.88 at 44M, and 0.11 to 1.00 at 78M. Thus the matched backbone can exploit additional test-time compute when it is expressed as generated tokens. The limitation is specific to the continuous-thought axis rather than to the underlying model. On standard Pythia models, where neither method uses recurrent pretraining, Coconut is likewise flat or declining relative to CoT at 410M–2.8B (§J). The design also contains a three-mode comparison on matched backbones: the $c{=}0$ column of Table 7 and the $T{=}1$ points of the depth ladder are the corresponding minimal-latent-compute baselines, and both curricula end in answer-only supervision, so the arms differ in how additional latent compute is added and share the training objective.

Mechanism. In continuous-thought mode, the visible reasoning chain is replaced by c continuous thoughts. Each thought runs the standard transformer on the current sequence and appends the last hidden state as the next input embedding rather than embedding a generated token, following Coconut (Hao et al., 2024). We implement this by allowing the transformer to consume precomputed embeddings.

Training. We use a staged curriculum following Coconut. At stage k, the first k reasoning steps are replaced by continuous thoughts, with one thought per replaced step, and only the final answer is supervised. At the final stage the entire reasoning chain is latent. We train for 300 steps per stage over six stages ($k{=}0\ldots5$), with batch size 16 and learning rate $8{\times}10^{-5}$. The final all-latent stage reaches a loss below 0.2, indicating that the comparison is between trained continuous-thought and recurrent-depth models rather than an untrained latent baseline.

Evaluation. We sweep $c \in \{0, 1, 2, 4, 8, 16, 32\}$ continuous thoughts. After inserting the thoughts following the question, we append the cue " The answer is" and greedily decode the answer on held-out problems with seed 999.

Results on the narrow depth range. On $K \leq 5$, additional continuous thoughts provide little systematic improvement. Table 7 contains all fifteen cells of the three-task, five-size design, averaged over three fine-tuning seeds. The best-over-c gain over $c{=}0$ ranges from 0.01 to 0.09, and no curve is monotone in c. These gains are substantially smaller than the $+0.47$ to $+0.65$ gains obtained by recurrent depth on the matched models.

The largest effect occurs on PermTrack at 19M, where $c{=}32$ improves accuracy by 0.09. The three seeds give $0.714/0.699/0.709$. Even here, however, the gain does not correspond to a clean depth-scaling effect: accuracy falls to 0.52 at $c{=}8$ before recovering, and the improvement at $c{=}32$ is nearly depth independent ($+0.10$ at $K{=}1$ versus $+0.09$ at $K{=}5$). The 121M gain is smaller and less robust, with one seed at 0.549 compared with 0.430 and 0.429 for the other two.

StateTrack shows the same pattern. The best-over-c gains are $+0.02$, $+0.03$, $+0.05$, $+0.03$, and $+0.02$ at 19, 30, 44, 78, and 121M, respectively, with best accuracies $0.43{\pm}0.02$, $0.42{\pm}0.05$, $0.51{\pm}0.04$, $0.40{\pm}0.08$, and $0.37{\pm}0.05$. Every size peaks at $c{=}8$ and then declines. The effect

Table 7: **Sequence-axis accuracy barely moves with the number of continuous thoughts c.** Accuracy on held-out problems for each ladder size and controlled task, fine-tuned with the Coconut curriculum from the standard-architecture twin of that rung. Each cell averages three fine-tuning seeds and the models are fine-tuned on $K \leq 5$. Δ is the best-over-c gain relative to c=0. None of the fifteen curves is monotone in c, and the largest gain here is 0.09, against +0.47 to +0.65 for the depth axis on the matched models. The same comparison on the wide-depth range is reported at the end of this section.

Task	Size	c=0	1	2	4	8	16	32	Δ
StateTrack	19M	0.41	0.42	0.42	0.39	0.43	0.42	0.39	+0.02
	30M	0.38	0.39	0.40	0.39	0.42	0.38	0.36	+0.03
	44M	0.46	0.47	0.47	0.47	0.51	0.48	0.45	+0.05
	78M	0.37	0.37	0.37	0.37	0.40	0.38	0.37	+0.03
	121M	0.35	0.36	0.35	0.36	0.37	0.35	0.34	+0.02
LogicTrack	19M	0.40	0.41	0.40	0.40	0.41	0.40	0.39	+0.01
	30M	0.46	0.46	0.46	0.46	0.47	0.46	0.46	+0.01
	44M	0.39	0.38	0.38	0.39	0.39	0.39	0.39	+0.01
	78M	0.41	0.41	0.41	0.41	0.42	0.41	0.39	+0.01
	121M	0.42	0.42	0.42	0.42	0.42	0.43	0.43	+0.01
PermTrack	19M	0.62	0.59	0.59	0.54	0.52	0.57	0.71	+0.09
	30M	0.64	0.64	0.64	0.64	0.63	0.64	0.66	+0.02
	44M	0.62	0.62	0.62	0.61	0.58	0.55	0.64	+0.03
	78M	0.53	0.53	0.52	0.51	0.52	0.49	0.55	+0.02
	121M	0.40	0.40	0.40	0.39	0.42	0.44	0.47	+0.06

is therefore small and non-monotone rather than the monotone saturation observed when compute is added through recurrent depth.

Stratifying by K does not reveal a hidden depth effect. At 44M, for example, K=5 remains between 0.33 and 0.41 across all c, while K=1 remains between 0.51 and 0.66. By comparison, recurrent depth improves every depth: at 19M, accuracy increases from 0.39 to 1.00 at K=1 and from 0.28 to 0.61 at K=5.

The relatively strong c=0 baseline partly explains the small aggregate gain. On these depths, many problems are already solved before additional continuous thoughts are introduced. The remaining gain becomes more visible when the task is made deeper, where the sequence axis can extend computation but does so less efficiently than recurrent depth.

Generalization beyond the training depth. We repeat the comparison on $K \in \{6, \ldots, 9\}$, outside the training range. Continuous thoughts improve over c=0 by +0.055, +0.115, and +0.095 at 19, 44, and 78M, respectively. The gains remain non-monotone in c and peak at c=8. Recurrent depth instead improves by +0.344, +0.461, and +0.575 at the same sizes and increases monotonically toward a plateau. The asymmetry therefore persists out of distribution.

Matched difficulty. The narrow range can underestimate the benefit of continuous thoughts because the task is close to saturation at larger model sizes. We therefore repeat the comparison on StateTrack with $K \leq 12$, using the same training data, three seeds, and five sizes as the recurrent-depth arm. Continuous thoughts now provide larger gains: +0.078, +0.082, +0.167, +0.146, and +0.069 at 19, 30, 44, 78, and 121M, respectively. Recurrent depth gains +0.33 to +0.62 at the same sizes.

The continuous-thought gain is also depth dependent, which bears on Claim 3. At 44M, the gain at the best c averages +0.08 over $K \leq 4$ and +0.23 over $K \geq 9$. Thus continuous thoughts can advance useful serial computation, but less efficiently than recurrent depth. The sequence curve peaks at c=16 and falls by 0.08 at c=32, whereas recurrent depth continues monotonically toward a plateau. Per unit of compute, the sequence axis returns between one eighth and one quarter of the gain of the depth axis at the same model size. We therefore do not interpret the result as evidence that continuous thoughts cannot advance circuit depth. They instead provide a substantially less efficient route to doing so.

J SEQUENCE AXIS ON PUBLIC BENCHMARKS: PROTOCOL AND FULL RESULTS

We next test whether the sequence-axis asymmetry of §5.2 is specific to the recurrent-pretraining setting. We therefore evaluate Coconut using its released training procedure on standard Pythia models and public benchmarks, where neither arm uses a recurrent core.

Protocol. For each Pythia-410M/1.4B/2.8B model and each benchmark (ProntoQA, ProsQA), we fine-tune two variants from identical pretrained weights and on the same training split:

1. *Continuous thoughts*, using the released Coconut curriculum: five epochs per stage, up to six latent stages, one continuous thought per replaced reasoning step, optimizer reset at each stage, and supervision on the tokens following the latent thoughts, for a total of 40 epochs; and
2. *CoT*, using standard autoregressive supervision on the complete visible reasoning chain.

We evaluate with a generation budget sufficient to emit the complete answer, using 256 tokens for CoT. All runs use learning rate 10^{-4}, bf16 precision, and an effective batch size of 32, held fixed across model sizes. Continuous-thought decoding is performed one example at a time, following the released Coconut evaluation procedure. Validation sets contain 500 problems each.

The effective batch size matters for Coconut, particularly at larger scale. At 2.8B, batch 64 gives 0.941–0.952 across seeds, whereas the matched batch-32 setting gives 0.772 (seeds 0.778 and 0.766). CoT is largely unchanged, improving slightly from 0.966 at batch 64 to 0.982 at batch 32. The same effect appears at 1.4B, where Coconut improves from 0.775 at batch 32 to 0.846 at batch 64. We therefore use batch 32 for the main comparison so that the training budget is matched across the ladder, and we report the batch-64 results as a sensitivity check.

Training and evaluation checks. Two implementation details materially affect the comparison.

First, the Coconut curriculum must reach the latent stages. The released training code introduces continuous thoughts at stage 1 (epoch 5), and a run that fails at this boundary can effectively revert to ordinary CoT training. We verify that every reported Coconut run passes all stage boundaries, uses continuous thoughts at evaluation, and completes the full 40-epoch schedule.

Second, the generation budget must be sufficient for CoT to finish its answer. With the default 128-token budget, ProntoQA CoT is artificially capped at 0.828: the 86 longest eight-step problems require roughly 130 tokens, causing the answer to be truncated. Increasing the budget to 256 removes the artifact and raises all three CoT models to 1.00. Continuous thoughts are unaffected because the reasoning steps remain latent. An insufficient generation budget would therefore unfairly favor continuous thoughts.

Optimization sensitivity. The two methods share pretrained weights, training data, optimizer, learning rate 10^{-4}, weight decay 0.01, bf16 precision, 40 epochs, and effective batch size 32. They differ only in whether the reasoning trajectory is supervised as visible tokens or replaced by continuous thoughts.

We also probe the main optimization parameters for Coconut. The effective batch size has the clearest effect: increasing it from 32 to 64 improves accuracy from 0.772 to 0.941–0.952 at 2.8B and from 0.775 to 0.846 at 1.4B. Halving or doubling the learning rate around 10^{-4} and increasing the per-stage training budget leave best accuracy within seed noise. The final latent stage is reached in every reported run. Thus the ordering below is not explained by the tested optimization choices, although larger batches can substantially improve the continuous-thought baseline.

ProntoQA. Both methods reach 1.00 at all three model sizes when CoT is evaluated with the 256-token budget. ProntoQA is therefore a positive control showing that continuous thoughts can solve the task, but the benchmark is too easy to distinguish the scaling behavior of the two axes.

ProsQA. ProsQA provides a more informative comparison because CoT remains below ceiling (Table 8). At the matched effective batch size of 32, Coconut versus CoT is $0.855/0.838$ at 410M, with standard deviations 0.002 and 0.011 over three and two seeds; $0.775/0.809$ at 1.4B, with

Table 8: **Sequence axis on ProsQA (best validation accuracy, 500 held-out problems) at a matched effective batch of 32 across the ladder.** On the harder, non-saturated benchmark, the continuous-thought variant does not out-scale CoT: it is within seed noise of it at 410M, falls behind at 1.4B, and at 2.8B CoT leads by a wide margin. All cells run the full 40-epoch curriculum (latent tokens verified active) at learning rate 10^{-4}, bf16. Coconut 410M/1.4B and CoT 1.4B average 2–3 seeds ($\pm$ values in text). Continuous thoughts are more batch-sensitive than CoT: at the larger effective batch 64 that fits 2.8B in memory, 2.8B Coconut recovers to 0.941 (still below CoT's 0.966), while CoT is essentially unchanged (§J).

	Pythia-410M	Pythia-1.4B	Pythia-2.8B
Continuous thoughts (Coconut)	0.855	0.775	0.772
CoT	0.838	0.809	0.982

standard deviations 0.031 and 0.015 over three seeds; and 0.772/0.982 at 2.8B, where Coconut averages two seeds (0.778 and 0.766) and CoT is single-seed.

At 410M, the +0.017 Coconut margin is smaller than the CoT seed spread. At larger scales, the ordering reverses and the gap widens. Thus continuous thoughts do not improve relative to CoT as model size increases. At the same time, Coconut remains substantially above the direct-answer baseline and reaches 0.941–0.952 at batch 64, showing that the latent sequence mechanism is functional even though it does not match CoT at larger scale.

The result differs from the original Coconut evaluation, which reports Coconut ahead of CoT on ProsQA (97.0 versus 77.5) with a GPT-2 base model (Hao et al., 2024). Our smallest model is consistent with that result: Coconut is at least competitive with CoT at 410M. At larger scales, however, CoT improves more rapidly, reversing the ordering. The advantage of a continuous-thought model at one scale therefore need not persist as the base model grows.

Together with the controlled-task results in §5.2, these experiments suggest that the sequence axis can support useful latent computation but does not provide the same scaling efficiency as recurrent depth on depth-bottlenecked reasoning.

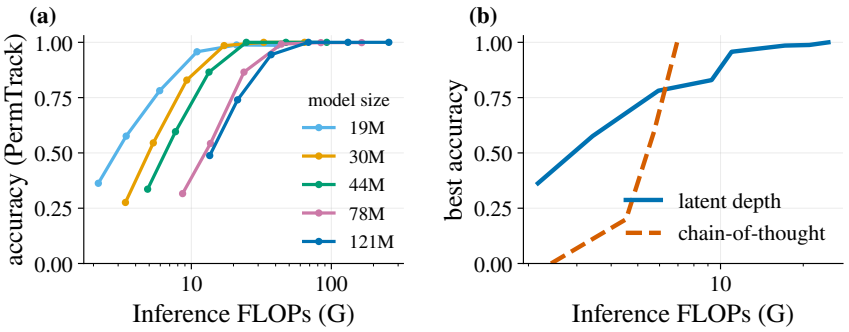


Figure 7: **Per-FLOP latent-vs-CoT on the second task (PermTrack). (a)** Latent-depth accuracy, three-seed mean, versus FLOPs per model size (color), rising and saturating as on StateTrack. **(b)** Efficient frontiers: latent depth (solid) leads CoT (dashed) over a wider budget range than on StateTrack, and CoT overtakes at 6.3×10^9 against 4.1×10^9 FLOPs, because a PermTrack chain carries a longer preamble and so fits a budget later. The crossover is set by the task's verbalization cost.

CoT baseline validity. The per-FLOP comparison uses the five sizes with standard-transformer twins. Each CoT baseline reproduces the gold chain whenever the token budget is sufficient. Accuracy therefore follows the chain-length CDF rather than model capability: at 128 tokens, all sizes reach 1.00 on both StateTrack and PermTrack, and their curves coincide to three decimals. We count an answer as correct only when the chain terminates within the budget. Otherwise a truncated chain can receive credit from its final digit even though the reasoning was not completed, inflating the low-budget accuracy by as much as 0.33.

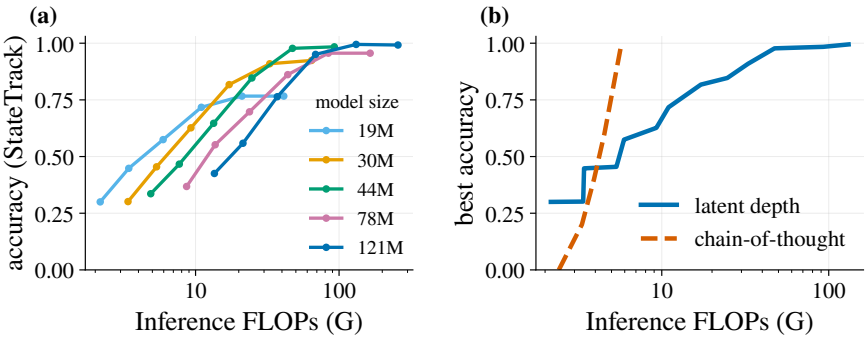


Figure 8: **Latent depth versus CoT at matched inference FLOPs (StateTrack). (a)** Latent-depth accuracy, three-seed mean, versus per-problem FLOPs per size (color). **(b)** Efficient frontiers for latent depth (solid) and CoT (dashed), five sizes each. Latent depth is efficient at low compute, and CoT overtakes near 4×10^9 FLOPs, the budget at which StateTrack chains begin to fit.

Self-consistency under matched FLOPs. Finally, we test whether the per-FLOP comparison changes when token compute is allocated across multiple sampled chains rather than a single greedy chain. On the same StateTrack CoT models, we sample N chains at temperature 0.8, with each chain limited to 64 tokens, and take a majority vote. Increasing N from 1 to 16 multiplies token-axis compute by $16\times$ but changes accuracy by exactly 0.000 at all five sizes, which remain at the greedy value of 0.600.

The absence of a gain is expected from the structure of the failures. At a 64-token budget, the failing problems are those whose gold chains do not fit within the budget. Every sampled chain encounters the same truncation, so additional samples cannot complete a chain that is individually too short. Self-consistency therefore reallocates compute across independent attempts without increasing the serial depth available to any one attempt. On these tasks, additional samples cannot substitute for additional reasoning steps, consistent with Proposition 2.

K HUGINN-3.5B: GSM8K ANCHOR, STATETRACK, AND THE FINE-TUNING NEGATIVE RESULT

This appendix reports the Huginn-3.5B experiments in §5.3. We first measure the depth law on GSM8K at the problem level, then evaluate few-shot latent recurrence on GSM8K, StateTrack, ProsQA, and SVAMP. We finally test fine-tuning and use the observed curves to recover a preliminary depth distribution.

The depth law on GSM8K. The aggregate GSM8K experiment uses three coarse difficulty buckets, but the deepest bucket contains only 39 problems, making its onset difficult to resolve. We therefore evaluate all 1319 test problems at every iteration count and stratify them after the sweep. We use the number of newline-separated reasoning lines in the gold solution as a structural depth proxy. Line counts are a proxy rather than a measured dependency depth, since the label-resampling measurement of §C requires resampling individual solution steps, which natural solutions do not expose. Six strata contain at least 40 problems, with $326/371/297/175/87/40$ examples at depths 2 through 7.

Accuracy decreases with depth at essentially every budget. Across all untied stratum pairs and iteration counts, the deeper stratum is less accurate in 119 of 146 comparisons (Kendall $\tau = 0.61$, $p < 10^{-14}$). At $T = 10$ the six strata achieve $0.42/0.30/0.23/0.15/0.13/0.08$, and the ordering remains strict at $T = 12$.

The plateaus reveal the reliability term of the law. By $T = 24$, every stratum has changed by at most 0.046 relative to $T = 16$. The resulting plateaus are $0.571/0.431/0.377/0.217/0.161/0.150$ from depths 2 through 7. Fitting $\log A_d$ against depth gives $R^2 = 0.961$ and $p = 0.747$. Thus the factor $A_d = p^d$ measured on the controlled ladder also describes the released Huginn model, although with substantially lower per-step reliability. This is consistent with the reasoning steps in GSM8K requiring more computation than the modular-arithmetic steps in the controlled tasks.

The onset itself does not separate across these strata. All six strata turn on between $T = 8$ and $T = 10$ under the half-rise convention, giving a slope of 0.00 reasoning lines per iteration with a bootstrap 95% interval of $[-0.71, 0.73]$. Because the plateaus are reached within the sweep, this is not a consequence of insufficient test-time compute. Instead, the low reliability of the released model suppresses the coverage transition that is visible on the controlled tasks. We therefore use GSM8K to test the reliability term, but not to estimate the depth-to-onset slope (§5.2). The six-stratum analysis also supersedes the earlier three-bucket analysis for this purpose: the coarse buckets provide too few distinct onsets to support a slope estimate.

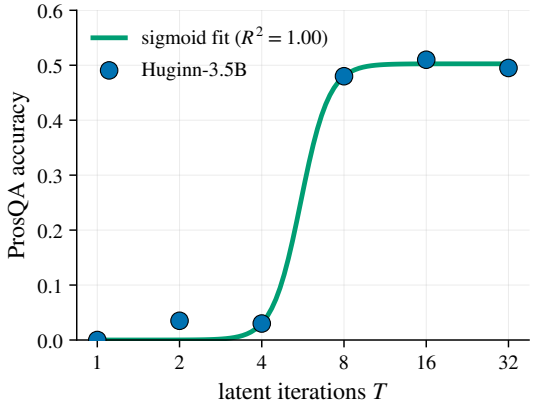


Figure 9: **The saturating law holds on a second real benchmark at 3.5B.** Released Huginn-3.5B on ProsQA (public deductive reasoning), evaluated few-shot with only the recurrence count T varied: accuracy is flat near zero until the reasoning depth is covered, rises sharply at $T \approx 8$, and plateaus, a saturating curve in $\log T$ (sigmoid R^2=0.996 versus log-linear 0.80). No training: recurrence depth is the released model's own `num_steps`.

GSM8K anchor. We first reproduce the aggregate GSM8K result on 400 problems. Huginn-3.5B achieves accuracies $0.00, 0.02, 0.01, 0.00, 0.08, 0.20, 0.34, 0.38$ at $T = 1, 2, 3, 4, 6, 8, 12, 16$, respectively. A sigmoid in $\log T$ gives $R^2 = 0.997$ (bootstrap 95% CI $[0.98, 1.00]$), compared with $R^2 = 0.76$ (CI $[0.73, 0.78]$) for the log-linear form. The bootstrap intervals are well separated, providing a clear preference for the saturating form.

We also stratify the problems by the number of reasoning steps in the gold solution: easy problems have at most 3 steps, medium problems have 4–5, and hard problems have more than 5. Accuracy is ordered by this depth proxy for every $T \geq 6$. This provides a structural difficulty measure that is less directly tied to surface prompt length than a token-count stratification. The finer six-stratum analysis above is used for the onset analysis.

Few-shot StateTrack. We next evaluate Huginn on StateTrack using a 4-shot prompt and the chat template. Accuracy is $0.044, 0.140, 0.188, 0.208$ at $T = 1, 2, 4, 8$, respectively. The gain is strongly depth dependent: at $T = 8$, accuracy is 0.40 for $K = 1$ and 0.06 for $K = 5$.

Few-shot ProsQA. On 200 held-out ProsQA problems (Figure 9), we use the same 4-shot protocol and greedy decoding, varying only the recurrence count. Accuracy is $0.00, 0.035, 0.03, 0.48, 0.51, 0.495$ at $T = 1, 2, 4, 8, 16, 32$. A sigmoid in $\log T$ gives $R^2 = 0.996$, compared with 0.80 for the log-linear form.

Few-shot SVAMP. On all 300 SVAMP test problems, accuracy is $0.02, 0.01, 0.02, 0.40, 0.61, 0.64$ at $T = 1, 2, 4, 8, 16, 32$. The sigmoid gives $R^2 = 0.998$ (bootstrap 95% CI $[0.99, 1.00]$), compared with 0.86 (CI $[0.83, 0.89]$) for the log-linear form. Easier problems also reach higher plateaus: problems with fewer arithmetic operations reach 0.68, compared with 0.53 for multi-operation problems, and turn on earlier.

The demonstrations contain only the final conclusion, "⟨name⟩ is a ⟨category⟩." They do not expose the intermediate reasoning chain. The model therefore must resolve the multi-hop deduction through

the latent recurrence rather than by imitating the demonstrated intermediate steps. Providing the full chain instead largely removes this distinction and substantially lowers the score.

Fine-tuning Huginn. We also attempted to fine-tune the released Huginn checkpoint on StateTrack. The training loss falls below 10^{-3} within 80 steps, but generation becomes degenerate for $T \geq 2$. The same behavior appears under both full fine-tuning and a lightweight adapter-plus-head variant with early stopping. Repeating the experiment with evaluation tokenization matched to training gives accuracy 0.088 at $T = 1$ and exactly 0 at every larger iteration count, ruling out a tokenization mismatch as the cause.

We therefore use the released checkpoint only as a qualitative cross-scale anchor and do not claim that the fitted law has been established for fine-tuned Huginn models. The failure appears to be a fine-tuning stability issue: the training loss collapses while generation degenerates only when recurrence is applied beyond the first step. Whether LoRA or other constrained adaptation methods can preserve the pretrained recurrent dynamics remains an open question.

Relation to probing analyses. Lu et al. (2025) report limited evidence for interpretable latent CoT in Huginn and only marginal gains from additional recurrence in their probing setting. These observations are compatible with our results. The law is a behavioral statement: an iteration must advance serial computation, but the intermediate state need not expose a directly decodable reasoning trace. The magnitude of the gain also depends on the task and the model's per-step reliability. Huginn improves from 0.00 to 0.38 on GSM8K between $T = 1$ and $T = 16$, but only from 0.04 to 0.21 on few-shot StateTrack between $T = 1$ and $T = 8$.

Recovering the depth distribution. We finally use the observed curves to obtain a preliminary estimate of the depth distribution on real tasks. Under

$$\mathrm{acc}(T) \approx \text{ceiling} \cdot F_{\mathcal{D}}(\alpha T),$$

the normalized increase in accuracy between successive iteration counts estimates the probability mass of problems whose required depth falls in the corresponding covered-depth interval.

On SVAMP, the recovered mass is concentrated between $T = 4$ and $T = 16$, consistent with its predominantly one- and two-operation problems. GSM8K places more mass at larger T, with the distribution extending from roughly $T = 6$ to $T = 16$, consistent with its longer multi-step solutions.

This inversion is qualitative because α and the absolute depth scale cannot be identified separately without an external depth annotation. The recovered distributions nevertheless preserve the expected ordering: GSM8K requires deeper computation than SVAMP.

L Recurrent-depth training cost and ladder feasibility

Recurrent-depth pretraining requires approximately one core pass per sampled iteration. With a mean recurrence of 32 and truncated backpropagation, this makes the effective training cost per token roughly an order of magnitude larger than that of a standard transformer with the same width.

On $8\times$H100, the 59M, 78M, and 121M models require approximately 7, 8, and 15 hours of training, respectively. This cost limits the from-scratch ladder to 121M parameters. We therefore use the released Huginn-3.5B checkpoint as a cross-scale anchor rather than extending the from-scratch ladder to billion-parameter models.