How Deep is Your Thinking? Evaluating the Depth & Efficiency of Long Horizon Reasoning

Anonymous authors
Paper under double-blind review

Abstract

Large language models (LLMs) with chain-of-thought reasoning capabilities have driven a step change in the classes of problems that are solvable with foundation models. In particular, the vanilla Transformer architecture is limited in the number of sequential computations it can perform within a fixed number of layers, whereas chain-of-thought reasoning greatly expands this capability in a manner that adapts to the problem complexity. However, the precise extent of this increase in capability, as well as the efficiency with which it is achieved, has not yet been systematically explored. To address this gap, we propose a new evaluation suite, DepthBench. DepthBench aims to measure the effective reasoning depth of models i.e. the number of sequential computations they are reliably capable of performing during inference. We focus on the domains of function composition for arithmetic and state tracking for planning, where we have direct control over problem complexity in terms of the required sequential computations necessary for success, allowing us to study how LLMs with and without chain-of-thought reasoning perform as a function of this complexity. We find that prominent open-source LLMs exhibit increases of one to three orders of magnitude in the achievable sequential computation depth, depending on the domain. We also find that models in a medium parameter range are the most efficient in their use of layers to increase depth, both with and without chain-of-thought. DepthBench enables the systematic evaluation of the effective depth of LLMs that employ adaptive computation and helps foster well-grounded innovation on the limits and efficiency of these models.

1 Introduction

While there have been a number of significant innovations since the rise to prominence of LLMs, the emergence of chain-of-thought reasoning models stands out because of the way that it expanded the class of problems that were actually solvable with foundation models. Of course, it is true that calling the intermediate tokens "thoughts" is a bit of a misnomer as neural networks are also capable of reasoning within their layers and are capable of some degree of "thinking" even without this augmentation. On the other hand, vanilla Transformer models were fundamentally limited in the amount of "thinking" that they were able to produce and chain-of-thought (CoT) models can theoretically augment this ceiling adaptively. As such, it is clear that CoT models have increased the amount of "thinking" LLMs can do and that this is theoretically only limited by compute. However, where we actually were without CoT and to what degree this ceiling has actually been lifted still remains murky. We release DepthBench for addressing exactly these kinds of questions about the limitations of reasoning depth and the efficiency with which it is achieved.

Limitations of Vanilla Transformers. The primary challenge of so called *multi-hop* or *multi-step* reasoning (Merrill & Sabharwal, 2025) is that multiple pieces of information must be considered in a way such that some information can only be properly integrated after earlier pieces of information have been understood. As such, a d-hop or d-step reasoning problem involves at the very least d sequential computations for it to be solved accurately. This immediately presents an issue for vanilla Transformer models as they can only perform at most L sequential computations where L is the number of layers in the network. A vanilla Transformer model clearly has an upper bound in its ability to perform multi-hop or multi-step reasoning and cannot perform it reliably when $d > L$. That said, even this analysis understates the severity of the issue. While a single layer and infinite width neural network can theoretically represent any function (Cybenko, 1989; Hornik et al., 1989; Lu

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.

et al., 2017), this is not the case for practical models with finite width layers (Nguyen et al., 2018). Moreover, it has been shown that neural networks with a greater number of layers can learn functions exponentially faster than shallower networks with fewer layers (Lin & Jegelka, 2018). So the reality is that to properly integrate a piece of information into a neural network's reasoning to the desired level of abstraction for a d-hop reasoning problem, $L > d$ layers will be needed in practice. Yet even this analysis still understates the problem for LLMs. Just because a Transformer has the needed number of layers, does not mean it will utilize its layers for that purpose, particularly if multi-hop or multi-step reasoning is not emphasized adequately during its pre-training process.

Expanded Capabilities of CoT Models. One chief advantage of CoT is in providing more expressive power that can be used for sequential computations (Li et al., 2024). Li et al. (2024) demonstrated that a N-step CoT model can solve any problem solvable by boolean circuits of size N. A Transformer model with L layers can then perform $L(N+1)$ sequential computations rather than just L. Moreover, if the model chooses when to stop thinking itself (as is often the case) this implies that it should have the expressive power to allocated more CoT tokens to solving a reasoning problem of any size d. That said, reasoning depth cannot necessarily be incremented with each additional token in practice. As a result, it is natural to question how many sequential operations are actually possible in a single pass through the network's L layers and how efficiently this is augmented with N more forward passes.

Releasing DepthBench. We have provided a new evaluation framework to the community that makes following contributions towards measuring the effective reasoning depth of foundation models:

- **DepthBench Framework:** We provide a framework for evaluating models across a suite of different problem complexities as measured by the number of sequential computations required to solve the problem. Specifically, we outline general recipes for creating problems of any depth in the domains of function composition and state tracking in Section 3.
- **Data Release:** We release a pilot instantiation of this framework with the task of arithmetic for function composition, and the Tower of Hanoi and 15 Puzzle tasks for state tracking.
- **New Metrics:** We propose novel metrics facilitated by this kind of data for measuring the effective depth of agents as well as the efficiency with which they achieve it in Section 4.
- **Empirical Findings:** We provide comprehensive experiments across prominent open source models with and without CoT using our pilot dataset in Section 5. This serves to demonstrate the new kinds of insights that can be acquired with our proposed framework and metrics.

Our data generation code is provided at `https://anonymous.4open.science/r/DepthBench/` and the data used in Section 5 is also provided at `https://dataverse.harvard.edu/previewurl.xhtml?token=b3678fbd-9968-4001-b24e-774598b0f68d`.

2 Related Work

Theoretical Depth of CoT Models. A line of theoretical work establishes that without intermediate tokens, transformers are computationally shallow. For example, log-precision transformers lie in TC^0 (a class of complexity problems) (Merrill & Sabharwal, 2023; Strobl et al., 2024); while constant-depth, constant-precision transformers cannot escape AC^0 (a weak complexity class) (Li et al., 2024). CoT prompting (Wei et al., 2022), foreshadowed by the rise of scratchpad models (Nye et al., 2021), lifts this computational ceiling. Merrill & Sabharwal (2024) and Li et al. (2024) prove that CoT length functions as a serial-computation budget, with polynomial CoT capturing P, while Feng et al. (2023) show that constant-size transformers fail at arithmetic and dynamic programming unless given CoT. These results predict a polynomial relationship between CoT length and achievable sequential depth, but the constants and effective rates have not been measured empirically before the introduction of DepthBench. Moreover, it has been shown that a fixed-length Transformer is computationally limited (equivalent to a finite automaton) and augmenting it with an external read-write memory (which CoT effectively provides via the context window) allows it to simulate a Universal Turing Machine (Schuurmans, 2023). Test-time compute scaling (Snell et al., 2024; Brown et al., 2024) and process reward models (Lightman et al., 2024) can then extend the effective depth even further at inference with the cost of more forward passes through all L layers.

Multi-Step Reasoning Benchmarks. Existing benchmarks probe multi-hop or multi-step reasoning indirectly. Multi-hop QA (Yang et al., 2018; Trivedi et al., 2022) and long-context suites such as BABILong (Kuratov et al., 2024) and RULER (Hsieh et al., 2024) embed hops in natural text, but are limited to small problem depths of 2 to 4 in practice. Meanwhile agentic benchmarks like SWE-bench (Jimenez et al., 2024), GAIA (Mialon et al., 2024), and Vending-Bench (Backlund & Petersson, 2025) test long-horizon coherence, but the necessary reasoning depth or horizon remains opaque, clouding the exact nature of performance gains. Closer to our setting, Dziri et al. (2023) and Valmeekam et al. (2023) vary computation-graph depth in compositional and planning tasks, and Kim & Schuster (2023) probes state tracking; however, these benchmarks neither parameterize depth nor isolate the effective depth improvements attributable to CoT. Moreover, long-context suites conflate context length with reasoning depth. DepthBench is designed to fill this gap, offering a controlled and fine-grained depth axis for arithmetic and planning state tracking, and a direct CoT vs. direct answer comparison at each depth.

Limits of CoT Models. A concurrent thread questions whether CoT delivers the depth it appears to. Indeed, faithfulness studies (Lanham et al., 2023; Turpin et al., 2023; Riemer et al., 2026) show that CoT may not be a directly interpretable driver of the answers derived. Furthermore, the "Illusion of Thinking" noted by Shojaee et al. (2025), together with Stechly et al. (2024), document an accuracy cliff in reasoning models past a problem complexity threshold. We see a similar limitation in our work where the length of the average CoT goes down as the model starts to "give up" on problems with a higher rate that have more than a particular depth (see Appendix C). In fact, we also consider the Tower of Hanoi domain as in (Merrill et al., 2024), but look at the state tracking problem rather than the more difficult and complicated issue of coming up with full solutions. This makes the analysis more directly interpretable as state tracking has a linear complexity while deriving full solutions to the problem has log-linear complexity and depends on the quality of your prior.

Attempts to Directly Measure Reasoning Depth. Concurrent work to ours also tries to measure LLM reasoning depth using the DeepMaze benchmark (Anonymous, 2026). Similar to (Merrill et al., 2024), this work considers coming up with a full action plan, which has higher than linear complexity in general. While this is helpful for understanding the horizon of solvable planning problems, the findings are thus disconnected from the notion of how many sequential computations the model is able to do. Chen et al. (2026) also look at the problem of quantifying LLM reasoning effort, but do so in terms of the CoT tokens that undergo significant revisions in deep model layers. This work is complementary to ours as this is not directly related to the number of sequential computations and cannot be readily used for understanding the effect of CoT in comparison to not using it at all.

Architectural alternatives for adding depth. A growing set of architectures aim to add computation depth without explicitly verbalizing CoT, and DepthBench provides a common axis on which to compare them to autoregressive CoT models in future studies. The most direct evidence that *effective depth, not verbal content, is what CoT supplies* comes from latent-reasoning work. Pfau et al. (2024) show that semantically empty filler tokens recover hard algorithmic capabilities, and Deng et al. (2024) distill explicit CoT into hidden-state depth, both supporting the framing that depth is the binding resource DepthBench should measure. Coconut (Hao et al., 2024) feeds hidden states recurrently, Quiet-STaR (Zelikman et al., 2024) trains internal thought tokens, and pause tokens (Goyal et al., 2024) interpose learned hidden vectors. A complementary line revives adaptive computation (Graves, 2016; Banino et al., 2021; Dehghani et al., 2019) and per-token routing (Raposo et al., 2024), while looped transformers (Saunshi et al., 2025) treat CoT itself as repeated forward-pass depth. Recursive architectures such as HRM (Wang et al., 2025) and the Tiny Recursive Model (Jolicoeur-Martineau, 2025) are explicitly designed to add effective depth without longer CoT, motivating the need for a benchmark that can fairly compare them to autoregressive reasoners on a common depth axis. Multi-agent decomposition offers an external alternative. Meyerson et al. (2025) complete a million-step Tower of Hanoi task with zero errors via atomic microagents. We view this as complementary to DepthBench's framing. Scaffolding can address depth at the system level, but DepthBench measures an intrinsic property of a model under any inference protocol. This matters for understanding architectural progress and for designing the components that any such scaffold ultimately relies on.

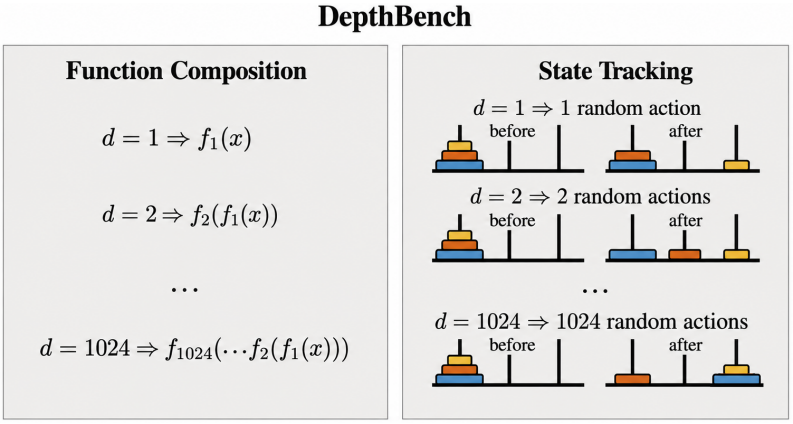

Figure 1: **The DepthBench Framework.** On the left we show how problems of varying depth d are generated in the function composition domain and on the right we show how problems of varying depth d are generated leveraging a given planning environment in the state tracking domain.

3 Construction of DepthBench

In this section we discuss the framework that underlies the creation of DepthBench in detail. As Figure 1 highlights, depth bench considered two families of problems that can be generated at any problem depth d. We outline the procedure for developing function composition problems in Section 3.2 and the procedure for developing state tracking problems in Section 3.3.

3.1 Problem Setting

In DepthBench, we consider problems from a particular task $t \in \mathcal{T}$ which require an exact sequential computation depth $d \in \mathbb{N}$ to solve. We desire that our tasks are defined such that any task t is defined for any problem complexity d so that individual input i and label ℓ pairs can be sampled $i, \ell \sim t(d)\ \forall t \in \mathcal{T},\ d \in \mathbb{N}$. Then we can estimate the probability that a particular model with a particular inference procedure θ produces the correct answer at depth d for task t as:

$$p_\theta(d \text{ is correct}) := \mathbb{E}_{i,\ell \sim t(d)}[\ell = p_\theta(i)]. \tag{1}$$

We can also estimate the average probability of being correct over a set of problem depths $\mathcal{D}$:

$$\bar{p}_\theta(\mathcal{D}) := \frac{1}{|\mathcal{D}|} \sum_{d \in \mathcal{D}} p_\theta(d \text{ is correct}). \tag{2}$$

In Section 4, we will explain in detail how these foundations enable various measurements of the effective depth k of the model and its particular inference procedure θ.

3.2 Function Composition Dataset

The first domain we consider in DepthBench is the general problem of function composition (see the left part of Figure 1). In this domain, problem inputs of depth d and task t are comprised of $i := \{t, x, f_1, ..., f_d\}$ where t is a description of the task, x is an input variable, and $f_1, ..., f_d$ are d functions that can be applied to a particular input. Then the label $\ell := f_d(...f_2(f_1(x)))$. In our experiments, the input i is translated into appropriate natural language text (see Appendix A for details on the specific prompts used). With some restrictions on the functions f_i considered, it is possible in this setup to make sure problem instances of task t require exactly d sequential computations.

Non-Parallelizable Function Compositions. The particular functions f_i chosen must be carefully selected such that when composed together they do not allow for any parallel computation and indeed require d sequential steps to produce. This can pose a problem when aspects of the problem can be easily factored or when specific symmetry relations hold as the functions are composed.

Addressing Guesses. There is also always some probability that a random guess or partially informed guesses can lead to a correct answer with some probability. The size of the label space should thus be

adjusted so that this probability is exceedingly small and labels that strongly align with the prior of the model being tested should be filtered out during data curation.

Arithmetic Instantiation. We ran into both of these issues while constructing our arithmetic dataset. When composing operations in random orders we found that the commutative and associative properties allowed for some degree of parallelization to be present in models. This is addressed by sequencing operations in a way that prevents these properties from being exploited. For example, every addition or subtraction operation is followed by a multiplication or floor division operation and vice versa. We also made sure that any labels inside the range $-10 \leq \ell \leq 10$ are filtered out, forcing a full regeneration of the problem instance. We found that LLMs were most likely to choose relatively small numbers when randomly guessing. We discuss the analysis that we performed to assure that sequential computations are performed without shortcuts skipping steps in Appendix C.

3.3 State Tracking Problems

The second domain we consider in DepthBench is the general problem of state tracking (see the right part of Figure 1, which uses the Tower of Hanoi environment as an example). State tracking can apply whenever we have access to a suitably complex deterministic environment $s' = \mathcal{M}_t(s, a)$, which is some function that produces an output state s' given an input state s and input action a. It is not a strict requirement that the environment is deterministic, but it is beneficial for having a higher signal to noise ratio in our experiments. In this domain, problem inputs of depth d and task t are comprised of $i := \{\mathcal{M}_t, s, a_1, ..., a_d\}$ where $\mathcal{M}_t$ is a description of the environment and prediction task, s is the initial state of the environment, and $a_1, ..., a_d$ are d randomly generated actions that can be applied to the environment. Then the label $\ell := \mathcal{M}_t(...\mathcal{M}_t(\mathcal{M}_t(s, a_1), a_2)..., a_d)$. In our experiments, the input i is again translated into appropriate natural language text (see Appendix A for details on the specific prompts used). With some restrictions on the environment and actions considered, it is again possible to make sure problem instances of task t require exactly d sequential computations.

Environment Complexity. The state tracking domain experiences similar issues to the function composition setting, but these are addressed by using nondegenerate environments of sufficient complexity. For example, the probability of randomly guessing an answer scales inversely with the size of the state space.[1] It is also important to make sure that the final state ℓ is not the same as the initial state s as LLMs have a tendency to randomly guess this. We chose the Tower of Hanoi and 15 Puzzle problems in our work as they are considered popular planning domains of significant complexity and difficulty. We also tried the Rubik's cube, but found that performance without CoT was too poor to facilitate the analysis we were pursuing as described in Section 4.

4 Measuring the Effective Reasoning Depth of LLMs

We now turn to the core motivation of this work, which is defining a measurement of the effective reasoning depth of a given model and inference procedure θ. We generally denote the concept of a model's reasoning depth as k and propose two specific ways of measuring it in Definitions 1 and 2. We also consider two notions of efficiency both in terms of the total layers traversed and in terms of floating point operations (FLOPs). We illustrate our motivation in Figure 2 where we compare a small Transformer with direct answers, a deeper Transformer with direct answers, and the deeper Transformer with CoT enabled. In this example, the effective reasoning depth is highest for the model with CoT, followed by the deeper Transformer with direct answers, with the smallest model having the lowest reasoning depth. However, in this specific example, we highlight that the efficiency per layer traversed shows the opposite trend. We will now discuss how to perform this kind of analysis.

4.1 Quantifying the Effective Reasoning Depth

Building off the quantities and notation established for the DepthBench setting in Section 3.1, we can define our first metric of the effective reasoning depth, which can be more precisely characterized as the *expected depth of the longest correct answer*. The question this metric attempts to answers is: if the model θ was asked to solve problems at every sequential complexity level $d \in \mathbb{N}$ for a task $t \in \mathcal{T}$,

[1] Note than in our Tower of Hanoi experiments we adjust probabilities in accordance with the 1/27 probability of randomly guessing a state. This can be interpreted as the probability above random of being correct.

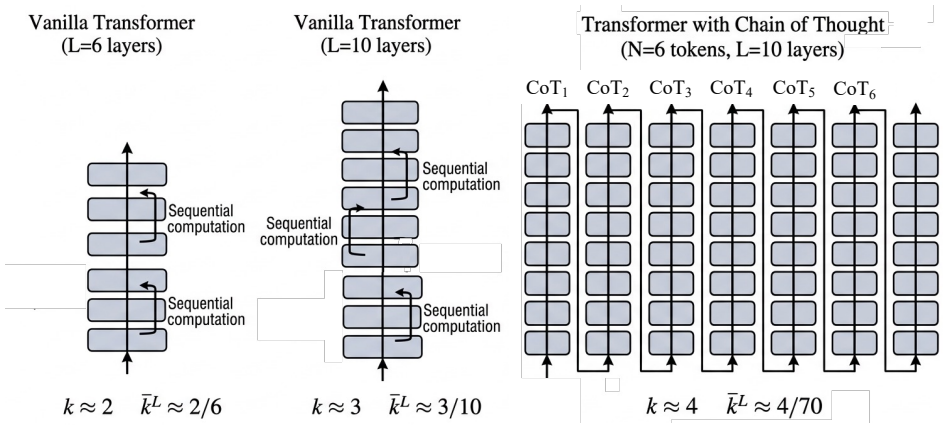

Figure 2: **Measuring Effective Depth and Efficiency.** We highlight three different LLM architecture patterns: a small vanilla Transformer (left), a larger vanilla Transformer (middle), and a Transformer augmented with Chain-of-Thought (right). We illustrate the effective depth of sequential computations k and the efficiency of these computations as a function of the total layers traversed $\bar{k}^L$.

what would is the expectation for the longest problem depth it will get right? We denote this metric as $\mathbb{E}[k]$ as it captures the idea of the expected effective reasoning depth of the model θ.

Definition 1 *(Expected Depth of Longest Correct Answer $\mathbb{E}[k]$): Consider a scenario where multiple answers, ordered by increasing depth $d \in \mathbb{N}$ for a task $t \in \mathcal{T}$ starting from 1, are generated sequentially one after another to infinity. For each specific depth d, there is a probability $p_\theta(d \text{ is correct})$ that the generated answer is correct for model θ. Let k be a random variable representing the depth of the longest correct answer found. Then:*

$$\mathbb{E}[k] = \sum_{d \in \mathbb{N}} \Big(d \times p_\theta(d \;\; is \;\; correct) \times p_\theta(none > d \;\; are \;\; correct) \Big) \tag{3}$$

Where $p_\theta(none > d \;\; are \;\; correct)$ is the probability that no correct answers are generated at depth $> d$. This probability is calculated using the following infinite product:

$$p_\theta(none > d \;\; are \;\; correct) = \prod_{j=d+1}^{\infty} \Big(1 - p_\theta(j \;\; is \;\; correct) \Big). \tag{4}$$

Perhaps the most appealing aspect of Definition 1 is that it provides a metric, which allows for fallibility in the model θ, that quantifies the effective depth without the need to set an arbitrary threshold of precision. The issue, on the other hand, is with the use of infinite sums. On the bright side, the infinite sum can be approximated by a finite sum if we are confident all higher values have 0 probability of being correct. For Transformers with direct answers, we can ensure this as long as we try all integers $d \in [1, L]$. We denote these empirical approximations of $\mathbb{E}[k]$ based on finite samples as $\hat{\mathbb{E}}[k]$. However, because models with CoT have no bound on the number of sequential computations they can perform, this metric is not suitable for analyzing their performance.

As a result, for a more computationally efficient metric of the effective reasoning depth for large depths, we turn to the use of a specific arbitrary precision threshold ϵ. Specifically, we can define the maximum depth at which ϵ precision is maintained as k_ϵ.

Definition 2 *(Maximum Depth at ϵ-Precision k_ϵ): For a model with a specific inference procedure θ we can define the maximum depth $d \in \mathbb{N}$ for a task $t \in \mathcal{T}$ where at least ϵ probability of the correct answer $p_\theta(d \text{ is correct})$ is maintained as:*

$$k_\epsilon = \max_{d \in \mathbb{N}} p_\theta(d \;\; is \;\; correct) \geq \epsilon. \tag{5}$$

While the metric is also not directly computable because of maximum operation over all integers, it can be approximated with a representative set of depths $d \in \mathcal{D}$ by leveraging scaling laws. Specifically, we want to make a predictive model $\hat{p}_\theta(d \;\; \text{is} \;\; \text{correct})$ as a function of d and find the value of problem depth d where $\hat{p}_\theta(d \;\; \text{is} \;\; \text{correct}) = \epsilon$, which we will denote as $\hat{k}_\epsilon$. In our experiments, we found that

a power law resulted in a better fit than linear regression, and that the best fit was from so called broken neural scaling laws (BNSL) (Caballero et al., 2022), which in particular do a better job of handling the region of low depth d where CoT models tend to get approximately perfect performance.

4.2 Efficiency Focused Metrics

Now that we have established two different ways of measuring k, we can talk about how to turn a specific measurement of k into an efficiency metric. We consider two types of efficiency in this paper:

Total layers traversed. One way to directly compare CoT models with models that answer directly is in terms of the total layers traversed as this represents the ceiling of the potential of the model for sequential computation. We denote this as $\bar{k}^L := k/[L_\theta \times (N + A)]$ where L_θ is the number of layers in the network, N is the number of CoT tokens, and A is the number of answer tokens. $L_\theta \times (N + A)$ can be collectively considered the total number of layers traversed.

Total computation. Another way to directly compare CoT models with models that answer directly is in terms of the total amount of computation. We denote this as $\bar{k}^F := k/F_\theta$ where F_θ is the number of floating point operations (FLOPs) expended by the network during its inference.

5 Empirical Findings

In this section, we provide comprehensive experiments across the function composition arithmetic dataset as well as the Tower of Hanoi and 15 Puzzle state tracking datasets for prominent open source models. For each problem depth d from 1 to 1024 we generate 1000 input i and label ℓ pairs following the procedure outlined in Section 3. For the direct answer models, we tested each across $d \in [1, 64]$ where $L = 64$ was the maximum number of layers in any model we tried. We tested the CoT models across depths $d \in \{1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024\}$. All of our experiments leveraged the given model hosted as a VLLM server with 8 H100 GPUs and workers sending them problems to solve (each leveraging 1 H100 GPU). The computation FLOP measurements are approximated based on the active parameter count for each model, the number of input tokens, and the number of generated tokens (Kaplan et al., 2020). See Appendix A for details about our prompting strategy.

Non-CoT Models. We tested the entire Qwen2.5-Instruct model family (Yang et al., 2025) from 0.5B to 32B. We also tried the base models, but found they had a hard time following directions to solve the problems. Additionally, we leveraged the Granite 4 model family (IBM Granite Team, 2025), which includes both dense, hybrid, and MoE models with the same number of layers. Our preliminary experiments with the OLMO-2 models (Groeneveld et al., 2024) yielded significantly worse results, so we omitted them to save space.

CoT Models. Our results for CoT models include the Qwen3 model family (Yang et al., 2026) from 0.6B to 32B, the Gemma 4 model family (Google DeepMind, 2026), and the GPT-OSS 20b and 120b models (Agarwal et al., 2025). Importantly, it is possible to turn off reasoning for Qwen3 and Gemma 4 models, but we still in some cases had difficulty extracting direct answers (without working through the problem first) for some of the Gemma 4 models. This makes sense as following our instruction to answer directly produces the wrong answer with high probability for high problem depths, so strong models may still decide to work out the result rather than follow our directions. Unfortunately, it was not possible to turn off CoT for GPT-OSS models. Additionally, we found that the reasoning effort level of GPT-OSS made little impact on the results and CoT lengths as a function of problem depth d.

5.1 Direct Answer Results without CoT

We provide results for our direct answer experiments in Table 1. It can be seen that function composition tends to be the easiest domain, followed by Tower of Hanoi, with 15 Puzzle consistently serving as the most difficult task. In general, bigger models with more depth seem to have the highest effective reasoning depth within each model family with the way that each model family is trained propelling some over others for reasons other than model size. As all models are trained specifically on math, arithmetic appears to be quite in distribution with all models achieving some degree of performance. However, sometimes this performance really does not generalize to domains such as 15 Puzzle as is the case for the smallest Granite 4 models. Interestingly, efficiency per layer often peaks for models that are smaller than the largest model. This make sense once the model reaches

Table 1: Direct answer performance across domains.

		Function Composition				Tower of Hanoi				15 Puzzle			
Model	L	$\hat{\mathbb{E}}[k]$	$\hat{\mathbb{E}}[\bar{k}^L]$	$\hat{k}_{5\%}$	$\hat{k}^L_{5\%}$	$\hat{\mathbb{E}}[k]$	$\hat{\mathbb{E}}[\bar{k}^L]$	$\hat{k}_{5\%}$	$\hat{k}^L_{5\%}$	$\hat{\mathbb{E}}[k]$	$\hat{\mathbb{E}}[\bar{k}^L]$	$\hat{k}_{5\%}$	$\hat{k}^L_{5\%}$
Qwen2.5-0.5B-Instruct	24	1.14	0.05	1.55	0.06	0.03	0.00	0.00	0.00	0.01	5.53e-04	0.00	0.00
Qwen2.5-1.5B-Instruct	28	1.00	0.04	2.10	0.07	0.01	0.00	0.00	0.00	0.01	4.67e-04	0.00	0.00
Qwen2.5-3B-Instruct	36	2.07	0.06	2.59	0.07	3.82	0.11	1.09	0.03	0.09	2.58e-03	1.06	0.03
Qwen2.5-7B-Instruct	28	3.14	0.11	3.27	0.12	0.73	0.03	2.69	0.10	0.18	6.28e-03	1.39	0.05
Qwen2.5-14B-Instruct	48	3.04	0.06	3.89	0.08	2.18	0.05	3.28	0.07	0.46	9.65e-03	1.77	0.04
Qwen2.5-32B-Instruct	64	4.03	0.06	4.24	0.07	3.29	0.05	5.96	0.09	0.71	1.11e-02	1.78	0.03
Qwen3-0.6B	28	2.49	0.09	1.52	0.05	0.01	0.00	0.00	0.00	0.00	0.00e+00	0.00	0.00
Qwen3-1.7B	28	2.88	0.10	2.69	0.10	0.09	0.00	1.18	0.04	0.03	9.97e-04	0.00	0.00
Qwen3-4B	36	2.26	0.06	2.85	0.08	1.57	0.04	3.33	0.09	0.19	5.35e-03	1.30	0.04
Qwen3-8B	36	3.36	0.09	2.92	0.08	1.57	0.04	3.22	0.09	0.24	6.67e-03	1.25	0.03
Qwen3-14B	40	2.80	0.07	4.26	0.11	1.52	0.04	3.89	0.10	0.58	1.44e-02	1.41	0.04
Qwen3-32B	64	3.82	0.06	4.22	0.07	1.89	0.03	4.66	0.07	0.65	1.01e-02	1.48	0.02
granite-4.0-micro	40	3.83	0.10	2.96	0.07	3.13	0.08	1.91	0.05	0.15	3.69e-03	1.22	0.03
granite-4.0-h-tiny	40	3.70	0.09	2.84	0.07	0.20	0.01	1.53	0.04	0.30	7.50e-03	1.53	0.04
granite-4.0-h-micro	40	3.12	0.08	2.99	0.07	0.27	0.01	1.36	0.03	0.11	2.83e-03	1.10	0.03
granite-4.0-h-small	40	3.29	0.08	3.45	0.09	2.19	0.05	3.07	0.08	0.57	1.43e-02	1.86	0.05
gemma-4-E2B-it	35	2.86	0.08	2.26	0.06	1.05	0.03	1.86	0.05	0.17	4.73e-03	1.65	0.05
gemma-4-E4B-it	42	2.63	0.06	3.00	0.07	1.37	0.03	2.56	0.06	0.43	1.02e-02	1.64	0.04
gemma-4-26B-A4B-it	30	—	—	—	—	—	—	—	—	—	—	—	—
gemma-4-31B-it	60	5.86	0.10	6.37	0.11	0.96	0.02	—	—	—	—	—	—

Table 2: Thinking model performance across domains.

		Function Composition				Tower of Hanoi				15 Puzzle			
Model	L	$\bar{p}_\theta$	$\hat{k}_{5\%}$	$\hat{k}^L_{5\%}$	N	$\bar{p}$	$\hat{k}_{5\%}$	$\hat{k}^L_{5\%}$	N	$\bar{p}_\theta$	$\hat{k}_{5\%}$	$\hat{k}^L_{5\%}$	N
Qwen3-0.6B	28	0.56	208.2	1.38e-3	5402	0.07	4.48	3.57e-5	4484	0.06	6.47	5.50e-5	4198
Qwen3-1.7B	28	0.75	590.5	1.81e-3	11638	0.30	22.75	1.99e-4	4083	0.27	16.61	1.26e-4	4714
Qwen3-4B	36	0.79	678.4	2.14e-3	8819	0.50	51.79	5.17e-4	2783	0.44	50.60	6.68e-4	2103
Qwen3-8B	36	0.82	898.8	2.73e-3	9139	0.54	107.1	9.39e-4	3167	0.51	133.1	3.48e-3	1061
Qwen3-14B	40	0.83	953.7	2.95e-3	8087	0.57	90.19	7.38e-4	3054	0.51	80.79	2.16e-3	936
Qwen3-32B	64	0.84	1036	1.98e-3	8180	0.56	114.6	8.61e-4	2079	0.50	104.7	6.58e-4	2486
gpt-oss-20b	24	0.70	923.1	8.58e-3	4481	0.66	210.8	1.08e-3	8126	0.63	176.1	1.34e-3	5464
gpt-oss-120b	36	0.87	1251	1.22e-2	2855	0.73	274.5	2.16e-3	3534	0.66	210.2	1.66e-3	3511
gemma-4-E2B-it	35	0.77	647.7	8.53e-3	2168	0.48	67.00	7.61e-04	2516	0.46	58.39	3.51e-4	4750
gemma-4-E4B-it	42	0.81	1043	5.25e-3	4732	0.54	131.76	1.06e-03	2961	0.53	65.14	2.82e-4	5507
gemma-4-26B-A4B-it	30	0.84	1202	1.81e-2	2208	0.81	329.0	6.09e-03	1800	0.63	213.2	1.63e-3	4347
gemma-4-31B-it	60	0.90	1366	8.20e-3	2776	0.78	332.4	5.05e-03	2369	0.68	258.5	8.09e-4	5327

beyond a critical size as the larger models within each family tend to be less over-trained as models are trained for roughly the same number of tokens. Unfortunately, the larger gemma models would not follow directions to produce direct answers in many cases (denoted by "—") despite their strong performance when they do provide direct answers relative to other comparable models. In Table 4 of Appendix B we provide the same table, but with efficiency based on FLOPs. Normalizing by FLOPs shows a much stronger bias towards smaller models than normalizing by layers, which makes sense due to the fact that layer width is also increased when the number of parameters is increased.

5.2 Results for Models with CoT Augmentation

We provide our results for models with CoT augmentation in Table 2. First of all, it is clear that we see a very significant increase in the effective reasoning depth across all domains. We again see a similar result for the Qwen3 family where models smaller than the 32B model tend to do better in terms of efficiency. Interestingly, some models of similar sizes across families can get away with a very different number of thinking tokens N across domains. For example, gemma-4-31B-it has less than half of the thinking tokens of Qwen3-32B for function composition whereas Qwen3-32B in turn uses less than half of the thinking tokens for 15 Puzzle. In Table 5 of Appendix B we again provide the same table, but with efficiency based on FLOPs. Once again we see that FLOP based metrics have an even stronger bias in favor for smaller model sizes. We provide an analysis of the content of the CoT to better understand the CoT length variability in these experiments in Appendix C.

External Correlation Study. To demonstrate the generality of our results, we performed a correlation study between $\hat{k}_{5\%}$ across our three domains and existing results for the same models on prominent reasoning benchmarks in Table 6 of Appendix B. First we should establish the level of internal consistence. The correlation between $\hat{k}_{5\%}$ estimates for function composition and Tower of Hanoi is 0.93, it is 0.86 between function composition and 15-Puzzle, and it is 0.95 between Tower of Hanoi and 15-Puzzle. For some prominent reasoning benchmarks we were able to find scores for all 12 models. For MMLU Pro the Spearman's correlation is 0.93 with the effective depth for function

Table 3: Comparing models with CoT with those that give direct answers.

		Function Composition			Tower of Hanoi			15 Puzzle		
Model	L	$M^{\hat{k}_{5\%}}$	$M^{\hat{\tilde{k}}^L_{5\%}}$	Corr(N, d)	$M^{\hat{k}_{5\%}}$	$M^{\hat{\tilde{k}}^L_{5\%}}$	Corr(N, d)	$M^{\hat{k}_{5\%}}$	$M^{\hat{\tilde{k}}^L_{5\%}}$	Corr(N, d)
Qwen3-0.6B	28	136.83	0.03	1.00	∞	∞	0.99	∞	∞	1.00
Qwen3-1.7B	28	219.54	0.02	1.00	∞	∞	0.82	∞	∞	1.00
Qwen3-4B	36	237.83	0.03	1.00	15.55	0.01	0.72	38.80	0.02	1.00
Qwen3-8B	36	307.76	0.03	1.00	33.21	0.01	0.92	106.08	0.10	1.00
Qwen3-14B	40	223.62	0.03	1.00	23.19	0.01	0.88	57.31	0.06	1.00
Qwen3-32B	64	245.21	0.03	1.00	24.61	0.01	1.00	70.79	0.03	1.00
gemma-4-E2B-it	35	287.00	0.13	1.00	35.94	0.01	0.92	35.39	0.01	1.00
gemma-4-E4B-it	42	347.83	0.07	1.00	51.47	0.02	0.94	39.73	0.01	1.00
gemma-4-31B-it	60	214.31	0.08	1.00	—	—	0.99	—	—	1.00

composition, 0.98 for Tower of Hanoi, and 0.91 for 15-Puzzle. For GPQA-Diamond the correlation is 0.90 for function composition, 0.94 for Tower of Hanoi, and 0.97 for 15-Puzzle. For LiveCodeBench the correlation is 0.88 for function composition, 0.84 for Tower of Hanoi, and 0.93 for 15-Puzzle. We also did our own evaluations on other relevant reasoning benchmarks (BFCLv3, IFBench, ZebraLogic, and HLE), finding strong statistically significant correlations with our effective depth measurements in each case. Finally, we found self-reported results for all models on the RULER 128k long context benchmark and found no statistically significant correlation with the effective reasoning depth for any of the DepthBench domains. This demonstrates that we have achieved our goal of designing a benchmark that isolates reasoning depth from challenges related to dealing with long contexts.

5.3 Comparing CoT Models with Direct Answer Models

Finally, in Table 3 we compare models for which we have results both with direct answers and with CoT augmentation. $M^{\hat{k}_{5\%}}$ denotes the multiplication in $\hat{k}_{5\%}$ with CoT compared to direct answers. This demonstrates that CoT results in increases in the effective reasoning depth that range from about 10x to about 300x depending on the model and domain. The infinite and missing values originate from 0 performance with direct answers in particular domains and missing direct answer data respectively. Meanwhile, $M^{\hat{\tilde{k}}^L_{5\%}}$ looks at the multiplication in layer efficiency of the effective reasoning depth with CoT. This demonstrates that these performance improvements are 30x to 100x less efficient at performing sequential computations per layer traversed than in the initial pass through the network with direct answers. This highlights that despite their performance improvements, there is still much untapped potential when it comes to efficiency with CoT models. We also plot Corr(N, d), which is the Spearman's correlation between the problem depth d and the length of correct answers in terms of CoT tokens. While these correlations are all very high, we do indeed see as in Merrill et al. (2024) that thinking length falls off for high complexity problems on average. However, for the case of answers that are actually correct, we see a monotonic increase with problem depth d. This result is expected as it perfectly aligns with the theory of where this additional reasoning depth comes from.

Robustness. We provide comprehensive confidence bands for all BSNL fits in Appendix D. We also wanted to show that changes to the arbitrarily chosen precision threshold have minimal impact on the results in Table 3. The multipliers in Table 3 range between 15.55 and 307.76 for a 5% threshold with the highest effective depth being 1366. When the threshold is 10% the multipliers range between 17.57 and 315.29 with the highest effective depth being 1089. Finally, when the threshold is 15% the multipliers range between 19.07 and 308.55 with the highest effective depth being 985.

6 Conclusion

In this paper we have proposed DepthBench, which provides a novel evaluation procedure for measuring the effective reasoning depth of various kinds of LLM models. Our framework provides general strategies for performing reasoning depth analysis in the domains of function composition and state tracking while providing a pilot study across the tasks of arithmetic, Tower of Hanoi, and 15 Puzzle. Our results demonstrate the remarkable gains of multiple orders of magnitude when using CoT to extend the effective reasoning depth. However, at the same time, our results suggest that these gains are many times less efficient during inference in achieving this depth than during the initial pass through the layers of the model. Providing researchers with systematic quantification of effective reasoning depth and its efficiency casts a novel perspective on the performance of reasoning models of all kinds. DepthBench thus serves to enable a wide variety of new research directions looking to either extend the limits of LLM reasoning depth or improve efficiency in achieving this depth.

AI USE STATEMENT

In this work, we used generative AI tools to help implement methods through coding to create datasets and evaluate models, develop code to generate synthetic data sets, reformat datasets, support qualitative and thematic data analysis, and interpret some results over unstructured data. We have not used generative AI tools to help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, propose or refine hypotheses, or provide feedback on research methodology. AI was not involved in the writing process of the main discourse. The rest of the required disclosure tasks are not applicable to this work. Additionally, we used generative AI tools to modify scientific figures and tables, and edit software code. We have reviewed all AI-assisted work. LLM-generated code was verified and tested for correctness by multiple authors. LLM-generated tables were checked against the raw data by the first author for quality control. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

We have made our datasets available while following best practices. Given the nature of the data, we do not foresee any immediate concerns related to discrimination, bias, or fairness.

REPRODUCIBILITY STATEMENT

We have provided detail on the LLM prompting strategy used in our experiments in Appendix A to facilitate reproducibility. We have also made the code available that we used to generate our data at `https://anonymous.4open.science/r/DepthBench/`. Additionally, we have made the actual data used in our experiments available at `https://dataverse.harvard.edu/previewurl.xhtml?token=b3678fbd-9968-4001-b24e-774598b0f68d`.

REFERENCES

S. Agarwal, L. Ahmad, J. Ai, S. Altman, A. Applebaum, et al. gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925*, 2025. URL https://arxiv.org/abs/2508.10925.

Anonymous. Deepmaze: A maze benchmark for quantifying reasoning depth in language models. In *Submitted to ACL Rolling Review - January 2026*, 2026. URL https://openreview.net/forum?id=nvMe9gCPfY. under review.

Axel Backlund and Lukas Petersson. Vending-bench: A benchmark for long-term coherence of autonomous agents. *arXiv preprint arXiv:2502.15840*, 2025.

Andrea Banino, Jan Balaguer, and Charles Blundell. PonderNet: Learning to ponder. *arXiv preprint arXiv:2107.05407*, 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.

Ethan Caballero, Kshitij Gupta, Irina Rish, and David Krueger. Broken neural scaling laws. *arXiv preprint arXiv:2210.14891*, 2022.

Wei-Lin Chen, Liqian Peng, Tian Tan, Chao Zhao, Blake JianHang Chen, Ziqian Lin, Alec Go, and Yu Meng. Think deep, not just long: Measuring llm reasoning effort via deep-thinking tokens. *arXiv preprint arXiv:2602.13517*, 2026.

George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2(4):303–314, 1989. doi: 10.1007/BF02551274.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers. In *International Conference on Learning Representations (ICLR)*, 2019. arXiv:1807.03819.

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.

Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jiang, Bill Yuchen Lin, Peter West, Chandra Bhagavatula, Ronan Le Bras, Jena D. Hwang, Soumya Sanyal, Sean Welleck, Xiang Ren, Allyson Ettinger, Zaid Harchaoui, and Yejin Choi. Faith and fate: Limits of transformers on compositionality. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. arXiv:2305.18654.

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. arXiv:2305.15408.

Google DeepMind. Gemma 4: Open models for advanced reasoning and efficiency. *Technical Report*, April 2026. URL https://storage.googleapis.com/deepmind-media/gemma/gemma-4-report.pdf.

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. arXiv:2310.02226.

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

Dirk Groeneveld, Iz Beltagy, Pete Walsh, Akshita Bhagia, et al. 2 olmo 2 furious. In *Proceedings of the Conference on Language Modeling (COLM)*, 2024. URL https://openreview.net/forum?id=2ezugTT9kU. Updated March 2025 for 32B release.

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.

Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989. doi: 10.1016/0893-6080(89)90020-8.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, and Boris Ginsburg. RULER: What's the real context size of your long-context language models? In *Conference on Language Modeling (COLM)*, 2024. arXiv:2404.06654.

IBM Granite Team. Granite 4.0 language models. IBM Research Technical Report, October 2025. URL https://huggingface.co/collections/ibm-granite/granite-40-language-models-67080597390977461 46c988c.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can language models resolve real-world GitHub issues? In *International Conference on Learning Representations (ICLR)*, 2024. arXiv:2310.06770.

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

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. URL https://arxiv.org/abs/2001.08361.

Najoung Kim and Sebastian Schuster. Entity tracking in language models. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2023. arXiv:2305.02363.

Yuri Kuratov, Aydar Bulatov, Petr Anokhin, Ivan Rodkin, Dmitry Sorokin, Artyom Sorokin, and Mikhail Burtsev. BABILong: Testing the limits of LLMs with long context reasoning-in-a-haystack. In *Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*, 2024. arXiv:2406.10149.

Tamera Lanham, Anna Chen, Ansh Radhakrishnan, Benoit Steiner, Carson Denison, Danny Hernandez, Dustin Li, Esin Durmus, Evan Hubinger, Jackson Kernion, et al. Measuring faithfulness in chain-of-thought reasoning. *arXiv preprint arXiv:2307.13702*, 2023.

Zhiyuan Li, Hong Liu, Denny Zhou, and Tengyu Ma. Chain of thought empowers transformers to solve inherently serial problems. In *International Conference on Learning Representations (ICLR)*, 2024. arXiv:2402.12875.

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. In *International Conference on Learning Representations (ICLR)*, 2024. arXiv:2305.20050.

Hongzhou Lin and Stefanie Jegelka. Resnet with one-neuron hidden layers is a universal approximator. *arXiv*, 2018. doi: 10.48550/arxiv.1806.10909.

Zhou Lu, Hongming Pu, Feicheng Wang, Zhiqiang Hu, and Liwei Wang. The expressive power of neural networks: A view from the width. *NIPS*, 2017. doi: 10.48550/arxiv.1709.02540.

William Merrill and Ashish Sabharwal. The parallelism tradeoff: Limitations of log-precision transformers. In *Transactions of the Association for Computational Linguistics (TACL)*, 2023. arXiv:2207.00729.

William Merrill and Ashish Sabharwal. The expressive power of transformers with chain of thought. In *International Conference on Learning Representations (ICLR)*, 2024. arXiv:2310.07923.

William Merrill and Ashish Sabharwal. A little depth goes a long way: The expressive power of log-depth transformers. *arXiv*, 2025. doi: 10.48550/arxiv.2503.03961.

William Merrill, Jackson Petty, and Ashish Sabharwal. The illusion of state in state-space models. In *International Conference on Machine Learning (ICML)*, 2024. arXiv:2404.08819.

Elliot Meyerson, Giuseppe Paolo, Roberto Dailey, Hormoz Shahrzad, Olivier Francon, Conor F. Hayes, Xin Qiu, Babak Hodjat, and Risto Miikkulainen. Solving a million-step LLM task with zero errors. *arXiv preprint arXiv:2511.09030*, 2025.

Grégoire Mialon, Clémentine Fourrier, Craig Swift, Thomas Wolf, Yann LeCun, and Thomas Scialom. GAIA: A benchmark for general AI assistants. In *International Conference on Learning Representations (ICLR)*, 2024. arXiv:2311.12983.

Quynh Nguyen, Mahesh Chandra Mukkamala, and Matthias Hein. Neural networks should be wide enough to learn disconnected decision regions. *arXiv*, 2018. doi: 10.48550/arxiv.1803.00094.

Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, Charles Sutton, and Augustus Odena. Show your work: Scratchpads for intermediate computation with language models. *arXiv preprint arXiv:2112.00114*, 2021.

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. arXiv:2404.15758.

David Raposo, Sam Ritter, Blake Richards, Timothy Lillicrap, Peter Conway Humphreys, and Adam Santoro. Mixture-of-depths: Dynamically allocating compute in transformer-based language models. *arXiv preprint arXiv:2404.02258*, 2024.

Matthew Riemer, Tommaso Tosato, Maximilian Puelma Touzel, Amin Memarian, Glen Berseth, Irina Rish, and Guillaume Dumas. Position: Collusion risks among ai reasoning agents justify certification requirements for making market decisions. *International Conference on Machine Learning*, 2026.

Nikunj Saunshi, Nishanth Dikkala, Zhiyuan Li, Sanjiv Kumar, and Sashank J. Reddi. Reasoning with latent thoughts: On the power of looped transformers. In *International Conference on Learning Representations (ICLR)*, 2025. arXiv:2502.17416.

Dale Schuurmans. Memory augmented large language models are computationally universal. *arXiv*, 2023. doi: 10.48550/arxiv.2301.04589.

Parshin Shojaee, Iman Mirzadeh, Keivan Alizadeh, Maxwell Horton, Samy Bengio, and Mehrdad Farajtabar. The illusion of thinking: Understanding the strengths and limitations of reasoning models via the lens of problem complexity. *arXiv preprint arXiv:2506.06941*, 2025.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

Kaya Stechly, Karthik Valmeekam, and Subbarao Kambhampati. Chain of thoughtlessness? an analysis of CoT in planning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. arXiv:2405.04776.

Lena Strobl, William Merrill, Gail Weiss, David Chiang, and Dana Angluin. What formal languages can transformers express? A survey. *Transactions of the Association for Computational Linguistics (TACL)*, 2024. arXiv:2311.00208.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. MuSiQue: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics (TACL)*, 2022. arXiv:2108.00573.

Miles Turpin, Julian Michael, Ethan Perez, and Samuel R. Bowman. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. arXiv:2305.04388.

Karthik Valmeekam, Matthew Marquez, Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. PlanBench: An extensible benchmark for evaluating large language models on planning and reasoning about change. In *Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*, 2023. arXiv:2206.10498.

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

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 (NeurIPS)*, 2022. arXiv:2201.11903.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2505.09388*, 2025. URL https://arxiv.org/abs/2505.09388.

An Yang, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. Qwen3 technical report: Scaling open foundation models. *arXiv preprint arXiv:2601.21337*, 2026. URL https://arxiv.org/abs/2601.21337.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018. arXiv:1809.09600.

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.

A Prompt Details

We attempted to perform as uniform of a prompting strategy as possible across models. For the specific "problem" prompts in each domain, see the data generation code provided at https://anonymous.4open.science/r/DepthBench/ and the actual data we used provided at https://dataverse.harvard.edu/previewurl.xhtml?token=b3678fbd-9968-4001-b24e-774598b0f68d.

Table 4: Direct answer performance across domains (with FLOP based efficiency).

		Function Composition				Tower of Hanoi				15 Puzzle			
Model	L	$\hat{\mathbb{E}}[k]$	$\hat{\mathbb{E}}[\bar{k}^F]$	$\hat{k}_{5\%}$	$\hat{\bar{k}}^F_{5\%}$	$\hat{\mathbb{E}}[k]$	$\hat{\mathbb{E}}[\bar{k}^F]$	$\hat{k}_{5\%}$	$\hat{\bar{k}}^F_{5\%}$	$\hat{\mathbb{E}}[k]$	$\hat{\mathbb{E}}[\bar{k}^F]$	$\hat{k}_{5\%}$	$\hat{\bar{k}}^F_{5\%}$
Qwen2.5-0.5B-Instruct	24	1.14	1.16e-09	1.55	1.58e-09	0.03	3.44e-11	0.00	0.00e+00	0.01	1.36e-11	0.00	0.00e+00
Qwen2.5-1.5B-Instruct	28	1.00	3.26e-10	2.10	6.82e-10	0.01	2.44e-12	0.00	0.00e+00	0.01	4.25e-12	0.00	0.00e+00
Qwen2.5-3B-Instruct	36	2.07	3.35e-10	2.59	4.20e-10	3.82	6.18e-10	1.09	1.76e-10	0.09	1.51e-11	1.06	1.72e-10
Qwen2.5-7B-Instruct	28	3.14	2.07e-10	3.27	2.15e-10	0.73	4.81e-11	2.69	1.77e-10	0.18	1.16e-11	1.39	9.11e-11
Qwen2.5-14B-Instruct	48	3.04	1.03e-10	3.89	1.32e-10	2.18	7.41e-11	3.28	1.11e-10	0.46	1.58e-11	1.77	6.04e-11
Qwen2.5-32B-Instruct	64	4.03	6.20e-11	4.24	6.52e-11	3.29	5.07e-11	5.96	9.17e-11	0.71	1.09e-11	1.78	2.74e-11
Qwen3-0.6B	28	2.49	2.08e-09	1.52	1.27e-09	0.01	8.19e-12	0.00	0.00e+00	0.00	0.00e+00	0.00	0.00e+00
Qwen3-1.7B	28	2.88	8.46e-10	2.69	7.91e-10	0.09	2.65e-11	1.18	3.47e-10	0.03	8.21e-12	0.00	0.00e+00
Qwen3-4B	36	2.26	2.82e-10	2.85	3.57e-10	1.57	1.96e-10	3.33	4.16e-10	0.19	2.41e-11	1.30	1.63e-10
Qwen3-8B	36	3.36	2.05e-10	2.92	1.78e-10	1.57	9.56e-11	3.22	1.97e-10	0.24	1.46e-11	1.25	7.65e-11
Qwen3-14B	40	2.80	9.45e-11	4.26	1.44e-10	1.52	5.13e-11	3.89	1.31e-10	0.58	1.95e-11	1.41	4.76e-11
Qwen3-32B	64	3.82	5.82e-11	4.22	6.44e-11	1.89	2.88e-11	4.66	7.10e-11	0.65	9.88e-12	1.48	2.25e-11
granite-4.0-micro	40	3.83	6.38e-10	2.96	4.93e-10	3.13	5.21e-10	1.91	3.18e-10	0.15	2.46e-11	1.22	2.04e-10
granite-4.0-h-tiny	40	3.70	1.85e-09	2.84	1.42e-09	0.20	1.00e-10	1.53	7.67e-10	0.30	1.50e-10	1.53	7.66e-10
granite-4.0-h-micro	40	3.12	5.20e-10	2.99	4.98e-10	0.27	4.53e-11	1.36	2.26e-10	0.11	1.89e-11	1.10	1.84e-10
granite-4.0-h-small	40	3.29	1.83e-10	3.45	1.92e-10	2.19	1.22e-10	3.07	1.71e-10	0.57	3.18e-11	1.86	1.03e-10
gemma-4-E2B-it	35	2.86	6.23e-10	2.26	4.91e-10	1.05	2.27e-10	1.86	4.05e-10	0.17	3.60e-11	1.65	3.59e-10
gemma-4-E4B-it	42	2.63	2.92e-10	3.00	3.33e-10	1.37	1.52e-10	2.56	2.84e-10	0.43	4.75e-11	1.64	1.82e-10
gemma-4-26B-A4B-it	30	—	—	—	—	—	—	—	—	—	—	—	—
gemma-4-31B-it	60	5.86	9.55e-11	6.37	1.04e-10	—	—	—	—	—	—	—	—

Table 5: Thinking model performance across domains (with FLOP based efficiency).

		Function Composition				Tower of Hanoi				15 Puzzle			
Model	L	$\bar{p}$	$\hat{k}_{5\%}$	$\hat{\bar{k}}^F_{5\%}$	N	$\bar{p}$	$\hat{k}_{5\%}$	$\hat{\bar{k}}^F_{5\%}$	N	$\bar{p}$	$\hat{k}_{5\%}$	$\hat{\bar{k}}^F_{5\%}$	N
Qwen3-0.6B	28	0.56	208	1.73e-7	5402	0.07	4.48	3.74e-9	4484	0.06	6.47	5.39e-9	4198
Qwen3-1.7B	28	0.75	591	1.74e-7	11638	0.30	22.8	6.69e-9	4083	0.27	16.6	4.89e-9	4714
Qwen3-4B	36	0.79	678	8.48e-8	8819	0.50	51.8	6.47e-9	2783	0.44	50.6	6.33e-9	2103
Qwen3-8B	36	0.82	899	5.48e-8	9139	0.54	107	6.53e-9	3167	0.51	133	8.12e-9	1061
Qwen3-14B	40	0.83	954	3.22e-8	8087	0.57	90.2	3.05e-9	3054	0.51	80.8	2.73e-9	936
Qwen3-32B	64	0.84	1036	1.58e-8	8180	0.56	115	1.75e-9	2079	0.50	105	1.60e-9	2486
gpt-oss-20b	24	0.70	923	1.28e-7	4481	0.66	211	2.93e-8	8126	0.63	176	2.45e-8	5464
gpt-oss-120b	36	0.87	1251	5.20e-9	2855	0.73	274	2.69e-8	3534	0.66	210	8.7e-10	3511
gemma-4-E2B-it	35	0.77	648	1.41e-7	2168	0.48	67.0	1.46e-8	2516	0.46	58.4	1.27e-8	4750
gemma-4-E4B-it	42	0.81	1043	1.16e-7	4732	0.54	132	1.46e-8	2961	0.53	65.1	7.24e-9	5507
gemma-4-26B-A4B-it	30	0.84	1202	2.31e-8	2208	0.81	329	4.33e-8	1800	0.63	213	2.81e-8	4347
gemma-4-31B-it	60	0.90	1366	2.22e-8	2776	0.78	332	8.02e-9	2369	0.68	258	4.21e-9	5327

For our CoT results we used a prompt template of: "{ problem } \n Return your final response within \\boxed{}." with no system prompt, turned enable reasoning on, and allowed for a maximum of 100k CoT tokens. We confirmed in our experiments that the correct answers always stayed within this budget even at the longest problem depth of 1024. For most of our direct answer experiments we used a prompt template of "{ problem } \n Return your final response within \\boxed{} and do NOT show step-by-step computation. For example, the output should be 'Answer: \\boxed' and nothing else. Any response with additional text will be considered invalid." with no system prompt. This was effective for most models, but for for Qwen3-0.6B, Qwen3-1.7B Qwen2.5-0.5B-Instruct, Qwen2.5-1.5B-Instruct, and the entire Gemma4 suite of models we also had to add a system prompt of "You must respond ONLY with 'Answer: \\boxed{ANSWER}' where ANSWER is the final numerical answer. Do not show any work, reasoning, or steps. Respond with ONLY the answer in the boxed format." We used a temperature 0.7 in all of our experiments.

B Additional Experiments

Here we provide additional tables to augment Tables 1 and 2 in the main text in which the efficiency metric is stated in terms of FLOPs rather than layers traversed. These results are provided in the form of Tables 4 and 5. We also provide the correlation of the effective depth metrics computed using DepthBench and performance on reasoning benchmarks in Table 6.

Table 6: Spearman's rank correlation of $\hat{k}_{5\%}$ and benchmark performance (p-values in parentheses).

Benchmark	Function Composition $\hat{k}_{5\%}$	Tower of Hanoi $\hat{k}_{5\%}$	15-Puzzle $\hat{k}_{5\%}$
MMLU Pro	0.93 (<0.01)	0.98 (<0.01)	0.91 (<0.01)
GPQA-Diamond	0.90 (<0.01)	0.94 (<0.01)	0.97 (<0.01)
LiveCodeBench	0.88 (<0.01)	0.84 (<0.01)	0.93 (<0.01)
BFCLv3	0.72 (<0.01)	0.60 (0.04)	0.63 (0.03)
IFBench	0.86 (<0.01)	0.91 (<0.01)	0.83 (<0.01)
ZebraLogic	0.75 (<0.01)	0.68 (0.03)	0.67 (0.02)
HLE	0.63 (0.04)	0.79 (<0.01)	0.70 (0.02)
RULER (128K)	0.52 (0.15)	0.63 (0.07)	0.50 (0.17)

C Analysis of the Chain-of-Thought Content

Is the Model Giving Up? Because our problem definition makes it easy for us to generate all intermediate solutions, we can verify programmatically that the CoT actually includes all intermediate solutions mentioned in the correct order within its CoT. This has allowed us to verify, for example, that every CoT model that ever gets 100% accuracy for a problem depth in any of our experiments also generates the full solution chain in order within its CoT in 100% of cases. As the reliability of answering correctly goes down, we also see the model in turn do this less often. We can keep track of the number of intermediate solutions skipped and can see this rise significantly, particularly at higher problem depths where accuracy is lower. Actually we find that at high problem depths all solutions that do not trace all intermediate solutions in order are guilty of skipping ahead, which leads to this dramatic loss of reliability. We believe that this skipping behavior can be seen as "giving up" as at lower depths models tend to even recheck their solutions a number times for greater reliability.

Variability Across Model Families. This behavioral analysis also helps explain the disparity between the number of CoT tokens across model families. For example, in the function composition domain at a problem depth of 4 we find that more verbose models from the Qwen3 family recheck intermediate solutions 6.56 times per problem step at 0.6B parameters and 8.41 times per problem at 32B parameters. Meanwhile, more concise models like Gemma4 still recheck intermediate solutions but less aggressively ranging from 3.52 times per problem step for gemma-4-31B-it to 4.59 time per problem step for gemma-4-E2B-it.

Control Showing Models Cannot Exploit Affine Subsequences. While in the main text we highlight the threat for our arithmetic dataset that commutativity and associativity present in allowing parallelizable shortcuts, there is also a concern we must rule out related to affine-subsequences. When multiplication is by sampled constants, any contiguous subsequence containing only addition, subtraction, and multiplication composes to a single affine map (A x + B). Equivalently, the sequence can be folded by composing (2×2) affine transformation matrices in logarithmic depth. Thus the irreducible depth of an instance is governed in principle not by the raw number of operations, but by the number and placement of genuinely non-collapsible operations. This is why the use of floor division is actually very important in our experiments, due to its non-linear behavior. We considered if superficial non-linearity should be added to the other operations of addition, subtraction, and multiplication, but found this to be unnecessary. Our analysis described above has allowed us to verify that there are 0 total cases where a CoT model gets the correct answer for function composition without computing each intended intermediate step in order. As a result, the point about affine-subsequences is of theoretical merit about what it may be possible for a model to do rather than something that actually materializes in our experiments. It is only possible to exploit this property if specific gates are learned for specific combinations of numbers in training, and it is simply not a justifiable use of parameters within the model to use so much of its capacity for specific mathematical relationships. We considered other alternatives where non-linearity is added to the operations of addition, subtraction, and multiplication, but felt that this makes the problems less intuitive and fundamental for humans. We also considered the idea of limiting the operations sampled to only floor divisions, but the issue is that the magnitude then constantly decreases meaning solutions will tend to drift towards zero, which makes guessing answers a very real possibility.

D ANALYSIS OF THE BNSL FITS

We summarize each accuracy-versus-depth curve by two scalars derived from a one-break broken neural scaling law (Caballero et al., 2022), the depth $\hat{k}_{5\%}$ at which fitted accuracy crosses 5%. This is a nonlinear functional of a fitted curve, so it does not inherit an error bar from the underlying accuracies automatically. This appendix quantifies how much of the estimate is supported by the data. We report goodness-of-fit statistics for every fit, bootstrap confidence intervals for the threshold, and the resulting intervals for the thinking-over-direct depth multipliers. Each cell below is one (model, domain, mode) triple, where mode is either direct answering or thinking with an explicit chain of thought. The direct condition is evaluated on a dense grid of consecutive depths, up to 32 points, and the thinking condition on a geometric grid of powers of two up to 1024, giving nine to eleven points.

D.1 FITTED MODEL

For each (model, domain, mode) cell we fit the one-break form

$$a(d) \;=\; a_0 + b\, d^{-c_0}\Big(1 + (d/d_1)^{1/f_1}\Big)^{-c_1 f_1}, \tag{6}$$

where d is problem depth and $a(d)$ is accuracy, by least squares over the depth grid. The threshold $\hat{k}_{5\%}$ solves $a(d) = 0.05 + a_{\text{base}}$ by Brent's method, where a_{base} is the chance baseline of the domain, which is $1/27$ for Tower of Hanoi and zero elsewhere.

D.2 BOOTSTRAP PROCEDURE

Each cell evaluates a finite number of items at each depth, so the measured accuracy at depth d is a binomial proportion rather than a known quantity. Near a 5% threshold this sampling noise is the dominant source of uncertainty, because a handful of additional successes at a shallow depth can move the crossing point substantially. We therefore resample at the level of individual items.

For each cell we draw $B = 1000$ bootstrap replicates. In replicate b the accuracy at each depth d is redrawn as $\tilde{a}_b(d) \sim \mathrm{Binomial}(n_d, \hat{a}(d))/n_d$, where n_d is the number of parsed items evaluated at that depth and $\hat{a}(d)$ the observed accuracy. Equation 6 is then refit to the resampled curve and $\hat{k}_{5\%}$ is recomputed from that refit. Refitting is what propagates fit uncertainty into the reported threshold, since a slightly different accuracy curve implies a different break location and hence a different threshold. All intervals are 95% percentile intervals over the replicates. Depth multipliers are ratios of two independently fitted thresholds. We pair bootstrap draws from the thinking and direct fits of the same cell and take percentiles of the resulting ratio distribution, so that both fits contribute to the reported interval.

D.3 THRESHOLD EXISTENCE

Two boundary cases require care, because in both the fitted curve has no crossing in the search interval and a naive implementation returns zero for each. We distinguish them explicitly. A cell is floored when fitted accuracy is already below the threshold at $d = 1$. The model fails at the shallowest depth evaluated, and $\hat{k}_{5\%}$ is reported as <1. A cell is right-censored when fitted accuracy remains above the threshold at the top of the search interval. The model has not yet failed at the deepest depth evaluated, so the data bound $\hat{k}_{5\%}$ from below but do not identify it.

D.4 GOODNESS OF FIT

Table 7 reports, for every cell, the total number of evaluated items (N), R^2, adjusted R^2, and the root mean squared residual in accuracy units. The fits are generally tight. Of the 54 cells with a defined R^2, 50 exceed 0.98 and the median is 0.9996, while residual RMSE stays below 0.02 accuracy units in 50 of the 55 cells, the largest being 0.046. Since Equation 6 has six free parameters, adjusted R^2 is the more honest summary for the sparser thinking cells, which are fit on nine to eleven depth points rather than the up-to-32 of the direct grid. It falls below 0.95 in four cells, discussed below.

Four cells fit poorly, and we flag them rather than average them away. Three are cases where the accuracy curve carries too little signal for a six-parameter break to be identified. Qwen3-0.6B on

Tower of Hanoi in the direct condition has $R^2 = 0.132$ and adjusted $R^2 = -0.077$, because its accuracy is flat noise around 2% across all 32 depths and never approaches the 8.7% Tower of Hanoi threshold, leaving the fit no better than a constant. Symmetrically, gemma-4-26B-A4B-it on Tower of Hanoi in the thinking condition has $R^2 = 0.323$ and adjusted $R^2 = -1.706$, because it holds accuracy at or near 1.0 across all nine depths evaluated. There is almost no variance to explain there, and the negative adjusted value reflects six parameters fit to a flat curve. Qwen3-1.7B on the 15-Puzzle in the direct condition is intermediate at $R^2 = 0.800$, with accuracy near zero and a weakly determined break. The fourth case, Qwen3-0.6B on the 15-Puzzle in the thinking condition, has $R^2 = 0.915$ but the largest residual in the table at $\mathrm{RMSE} = 0.046$ on eleven points.

In each case the poor fit is a property of the underlying curve rather than of the fitting procedure, and the qualitative reading is unambiguous. The flat-low cells do not solve the task at any depth tested, and the flat-high cell has not failed by the deepest depth evaluated. A fifth cell, Qwen3-0.6B on the 15-Puzzle in the direct condition, scores exactly zero at every depth, leaving R^2 undefined. Its floored $\hat{k}_{5\%}$ is the meaningful statement.

D.5 Confidence intervals for $\hat{k}_{5\%}$

Table 8 reports the point estimates with their bootstrap intervals, split across two panels for legibility. Intervals for the direct condition are narrow throughout, typically a few tenths of a depth unit, reflecting the large item counts of up to 32,000 per cell and the shallow thresholds involved. Intervals for the thinking condition are wider in both absolute and relative terms, for two reasons. The thinking grid is geometric and so constrains the curve at fewer points, and the thresholds themselves are one to three orders of magnitude deeper, where the fitted curve is flatter and a small vertical perturbation displaces the crossing further. The comparison of interest survives this noise comfortably. For every model with both conditions measured, the thinking interval for $\hat{k}_{5\%}$ lies entirely above the direct interval, with the two separated by more than an order of magnitude in the Function Composition domain.

D.6 Depth multipliers

Table 9 gives the ratio $M^{\hat{k}_{5\%}} = \hat{k}^{\text{think}}_{5\%} / \hat{k}^{\text{direct}}_{5\%}$ with its paired bootstrap interval. The multipliers span roughly one to three orders of magnitude and are ordered consistently across domains. Function Composition admits the largest gain from chain-of-thought, the 15-Puzzle an intermediate one, and Tower of Hanoi the smallest. The intervals are wide in absolute terms for the larger models, a consequence of dividing by a small direct threshold that is itself estimated precisely but sits close to the shallowest depth evaluated.

Table 7: Goodness-of-fit statistics for the one-break BNSL fits.

Model	Domain	Mode	N	R^2	adj. R^2	RMSE
Qwen3-0.6B	Function Composition	Direct	17,059	1.000	1.000	0.0022
	Function Composition	Thinking	10,590	0.999	0.999	0.0101
	Tower of Hanoi	Direct	31,910	0.132	-0.077	0.0044
	Tower of Hanoi	Thinking	6,182	0.983	0.957	0.0172
	15-Puzzle	Direct	31,920	—	—	0.0000
	15-Puzzle	Thinking	5,733	0.915	0.787	0.0463
Qwen3-1.7B	Function Composition	Direct	31,997	1.000	1.000	0.0012
	Function Composition	Thinking	10,661	0.999	0.999	0.0088
	Tower of Hanoi	Direct	2,037	0.989	0.986	0.0226
	Tower of Hanoi	Thinking	5,875	0.999	0.999	0.0085
	15-Puzzle	Direct	32,000	0.800	0.752	0.0013
	15-Puzzle	Thinking	5,481	1.000	1.000	0.0027
Qwen3-4B	Function Composition	Direct	31,993	1.000	1.000	0.0022
	Function Composition	Thinking	10,846	1.000	1.000	0.0010
	Tower of Hanoi	Direct	31,837	0.991	0.989	0.0105
	Tower of Hanoi	Thinking	6,363	0.999	0.999	0.0104
	15-Puzzle	Direct	31,823	1.000	1.000	0.0002
	15-Puzzle	Thinking	5,195	1.000	0.999	0.0080
Qwen3-8B	Function Composition	Direct	31,993	1.000	1.000	0.0023
	Function Composition	Thinking	10,918	1.000	1.000	0.0045
	Tower of Hanoi	Direct	32,000	0.990	0.988	0.0125
	Tower of Hanoi	Thinking	6,755	0.995	0.986	0.0320
	15-Puzzle	Direct	30,502	1.000	1.000	0.0003
	15-Puzzle	Thinking	3,640	0.994	0.986	0.0324
Qwen3-14B	Function Composition	Direct	31,739	1.000	1.000	0.0021
	Function Composition	Thinking	10,931	1.000	0.999	0.0044
	Tower of Hanoi	Direct	31,997	0.999	0.999	0.0044
	Tower of Hanoi	Thinking	7,332	1.000	1.000	0.0061
	15-Puzzle	Direct	31,998	1.000	1.000	0.0007
	15-Puzzle	Thinking	3,744	1.000	1.000	0.0049
Qwen3-32B	Function Composition	Direct	31,925	1.000	1.000	0.0023
	Function Composition	Thinking	10,956	1.000	1.000	0.0023
	Tower of Hanoi	Direct	32,000	0.996	0.995	0.0092
	Tower of Hanoi	Thinking	5,584	0.999	0.999	0.0103
	15-Puzzle	Direct	31,868	0.999	0.999	0.0024
	15-Puzzle	Thinking	5,548	1.000	0.999	0.0087
gemma-4-E2B-it	Function Composition	Direct	31,987	1.000	1.000	0.0023
	Function Composition	Thinking	4,333	0.996	0.989	0.0235
	Tower of Hanoi	Direct	31,700	0.983	0.979	0.0067
	Tower of Hanoi	Thinking	6,971	0.998	0.996	0.0175
	15-Puzzle	Direct	31,604	1.000	1.000	0.0000
	15-Puzzle	Thinking	7,959	0.999	0.999	0.0103
gemma-4-E4B-it	Function Composition	Direct	31,950	1.000	1.000	0.0025
	Function Composition	Thinking	1,195	0.999	0.998	0.0078
	Tower of Hanoi	Direct	24,628	0.987	0.983	0.0136
	Tower of Hanoi	Thinking	7,767	0.999	0.997	0.0151
	15-Puzzle	Direct	31,965	1.000	1.000	0.0012
	15-Puzzle	Thinking	7,742	1.000	0.999	0.0071
gemma-4-26B-A4B-it	Function Composition	Thinking	9,183	1.000	0.999	0.0051
	Tower of Hanoi	Thinking	2,838	0.323	-1.706	0.0187
	15-Puzzle	Thinking	4,449	0.999	0.996	0.0169
gemma-4-31B-it	Function Composition	Direct	10,497	0.998	0.997	0.0146
	Function Composition	Thinking	9,756	1.000	1.000	0.0007
	Tower of Hanoi	Thinking	4,920	1.000	0.999	0.0064
	15-Puzzle	Thinking	6,410	1.000	1.000	0.0009

Table 8: Bootstrap 95% confidence intervals for the BNSL threshold.

Model	Domain	Mode	$\hat{k}_{5\%}$	95% CI
Qwen3-0.6B	Function Composition	Direct	1.5	[1.3, 1.7]
	Function Composition	Thinking	208.2	[186.4, 230.9]
	Tower of Hanoi	Direct	<1	—
	Tower of Hanoi	Thinking	4.5	[3.1, 4.7]
	15-Puzzle	Direct	<1	—
	15-Puzzle	Thinking	6.5	[2.7, 6.7]
Qwen3-1.7B	Function Composition	Direct	2.7	[2.5, 2.8]
	Function Composition	Thinking	590.5	[560.8, 617.8]
	Tower of Hanoi	Direct	2.6	[2.6, 2.7]
	Tower of Hanoi	Thinking	22.7	[21.1, 25.8]
	15-Puzzle	Direct	<1	—
	15-Puzzle	Thinking	16.6	[15.5, 82.5]
Qwen3-4B	Function Composition	Direct	2.9	[2.7, 3.0]
	Function Composition	Thinking	678.4	[631.1, 740.0]
	Tower of Hanoi	Direct	3.3	[3.0, 3.6]
	Tower of Hanoi	Thinking	51.8	[43.9, 64.2]
	15-Puzzle	Direct	1.3	[1.2, 1.7]
	15-Puzzle	Thinking	50.6	[37.3, 60.0]
Qwen3-8B	Function Composition	Direct	2.9	[2.8, 3.1]
	Function Composition	Thinking	898.8	[866.7, 918.9]
	Tower of Hanoi	Direct	3.2	[3.0, 3.5]
	Tower of Hanoi	Thinking	107.1	[91.7, 219.6]
	15-Puzzle	Direct	1.3	[1.1, 1.5]
	15-Puzzle	Thinking	133.1	[116.8, 289.3]
Qwen3-14B	Function Composition	Direct	4.3	[4.1, 4.5]
	Function Composition	Thinking	953.7	[932.5, 983.3]
	Tower of Hanoi	Direct	3.9	[3.6, 4.1]
	Tower of Hanoi	Thinking	90.2	[82.7, 109.4]
	15-Puzzle	Direct	1.4	[1.3, 1.5]
	15-Puzzle	Thinking	80.8	[63.6, 108.3]
Qwen3-32B	Function Composition	Direct	4.2	[4.0, 4.4]
	Function Composition	Thinking	1035.8	[1001.6, 1088.2]
	Tower of Hanoi	Direct	4.7	[4.4, 4.9]
	Tower of Hanoi	Thinking	114.6	[93.9, 151.1]
	15-Puzzle	Direct	1.5	[1.4, 1.6]
	15-Puzzle	Thinking	104.7	[83.3, 128.4]
gemma-4-E2B-it	Function Composition	Direct	2.3	[2.2, 2.4]
	Function Composition	Thinking	647.7	[568.6, 743.1]
	Tower of Hanoi	Direct	1.9	[1.6, 2.1]
	Tower of Hanoi	Thinking	67.0	[60.0, 79.8]
	15-Puzzle	Direct	1.6	[1.3, 1.7]
	15-Puzzle	Thinking	58.4	[55.4, 61.2]
gemma-4-E4B-it	Function Composition	Direct	3.0	[2.8, 3.1]
	Function Composition	Thinking	1043.0	[733.2, 1388.4]
	Tower of Hanoi	Direct	2.6	[2.3, 2.9]
	Tower of Hanoi	Thinking	131.8	[114.9, 152.2]
	15-Puzzle	Direct	1.6	[1.5, 1.7]
	15-Puzzle	Thinking	65.1	[62.8, 67.2]
gemma-4-26B-A4B-it	15-Puzzle	Thinking	213.2	[181.3, 265.0]
gemma-4-31B-it	Function Composition	Direct	6.4	[6.1, 6.6]
	Function Composition	Thinking	1366.0	[1085.4, 1586.2]
	15-Puzzle	Thinking	258.5	[225.6, 302.2]

Table 9: Depth multipliers $M^{\hat{k}_{5\%}}$ with paired bootstrap intervals.

Model	Domain	$M^{\hat{k}_{5\%}}$	95% CI
Qwen3-0.6B	Function Composition	136.8	[119.6, 165.7]
Qwen3-1.7B	Function Composition	219.5	[204.3, 235.9]
	Tower of Hanoi	8.7	[8.1, 9.8]
Qwen3-4B	Function Composition	237.8	[217.2, 265.1]
	Tower of Hanoi	15.6	[13.0, 19.6]
	15-Puzzle	38.8	[26.7, 46.5]
Qwen3-8B	Function Composition	307.8	[288.8, 326.6]
	Tower of Hanoi	33.2	[27.7, 71.3]
	15-Puzzle	106.1	[85.8, 240.3]
Qwen3-14B	Function Composition	223.6	[213.2, 236.8]
	Tower of Hanoi	23.2	[21.1, 28.6]
	15-Puzzle	57.3	[44.1, 78.5]
Qwen3-32B	Function Composition	245.2	[232.8, 262.0]
	Tower of Hanoi	24.6	[20.1, 32.3]
	15-Puzzle	70.8	[56.1, 88.5]
gemma-4-E2B-it	Function Composition	287.0	[247.5, 326.3]
	Tower of Hanoi	35.9	[30.5, 43.7]
	15-Puzzle	35.4	[32.7, 46.8]
gemma-4-E4B-it	Function Composition	347.8	[246.1, 463.1]
	Tower of Hanoi	51.5	[43.3, 62.2]
	15-Puzzle	39.7	[36.9, 43.0]
gemma-4-31B-it	Function Composition	214.3	[168.0, 259.8]