Thought calibration: Efficient and confident test-time scaling

Menghua Wu, Cai Zhou, Stephen Bates, Tommi Jaakkola
Department of Computer Science
Massachusetts Institute of Technology
Cambridge, MA 02319

Abstract

Reasoning large language models achieve impressive test-time scaling by thinking for longer, but this performance gain comes at significant compute cost. Directly limiting test-time budget hurts overall performance, but not all problems are equally difficult. We propose *thought calibration* to decide dynamically when thinking can be terminated. To calibrate our decision rule, we view a language model's growing body of thoughts as a nested sequence of reasoning trees, where the goal is to identify the point at which novel reasoning plateaus. We realize this framework through lightweight probes that operate on top of the language model's hidden representations, which are informative of both the reasoning structure and overall consistency of response. Based on three reasoning language models and four datasets, thought calibration preserves model performance with up to a 60% reduction in thinking tokens on in-distribution data, and up to 20% in out-of-distribution data.[1]

1 Introduction

Test-time scaling presents a new paradigm for improving language model reasoning by expending large amounts of compute during inference (Kaplan et al., 2020; Wei et al., 2022). Though the strategies for eliciting reasoning vary – from large-scale reinforcement learning (Guo et al., 2025a) to explicit tree search (Zhang et al., 2024b,a) – a common effect is that language models improve by sampling substantially more tokens. This may result in wasted compute on easy problems (Chen et al., 2024; Sui et al., 2025), but naively limiting the generation length leads to pronounced drops in accuracy (Muennighoff et al., 2025). This motivates early stopping strategies that reduce the inference budget without significantly degrading performance, and control the extent of impact, if performance must be compromised.

Numerous methods have been proposed for teaching language models to be economical with their token budgets (Han et al., 2024; Arora and Zanette, 2025; Sui et al., 2025), or for identifying opportune stopping points (Yang et al., 2025; Zhang et al., 2025). While these methods demonstrate strong empirical performance, they lack strict statistical guarantees about when they could fail. In an orthogonal direction, conformal prediction has been adapted to equip language models with calibrated confidences about the quality or consistency of their generations (Mohri and Hashimoto, 2024; Quach et al., 2024; Rubin-Toles et al., 2025a,b; Cherian et al., 2024). However, most of these algorithms operate through post-hoc filtering and require external LLM-based validation for scoring intermediate steps – rendering them unsuitable for actively terminating generation.

In this work, we jointly pursue an effective and calibrated decision rule to determine when a language model can stop "thinking." To do so, we introduce the notion of a reasoning tree, where at each step of sampling, a language model either adds a new leaf, walks along the tree, or backtracks to a previous step. Notably, identifying when the thoughts have converged is equivalent to detecting when this reasoning tree stops growing. Inspired by this concept, we approach early stopping as multiple hypothesis testing problem. At each generation step, we test whether the current tree is expected to change, based on the predictions of lightweight probes over the language model's hidden representations. Our algorithm is based on the Learn then Test framework (Angelopoulos et al., 2021), which provides finite-sample, distribution-free guarantees for controlling the risk of our decisions.

We evaluate this strategy, *thought calibration*, based on its ability to guide efficient reasoning, and whether its decisions are well-calibrated. Our experiments consider two empirical settings: we may or may not have access to training and cali-

---

[1] Code and data are available at: `https://github.com/rmwu/thought-calibration`

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 14291–14305
November 4-9, 2025 ©2025 Association for Computational Linguistics

bration data from the true test distribution. In the first setting, we train variants of thought calibration using three reasoning lanugage models (DeepSeek-R1 distilled Qwen 32B and Llama 70B (Guo et al., 2025a; Yang et al., 2024; Grattafiori et al., 2024), QwQ 32B (Team, 2025)), evaluated on a helf-out split of s1K-1.1 (Muennighoff et al., 2025). Here, we are able to halve the number of thinking tokens across accuracy levels, with a **maximum reduction of 60%**. Then we evaluate Qwen 32B-based thought calibration on three test datasets: AIME 24, GPQA Diamond (Rein et al., 2024), MATH-500 (Lightman et al., 2023)). Though these datasets vary in format and difficult, thought calibration is still able to reach **up to a 20% reduction in thinking tokens**, and in the worst case, is **always as efficient as naive budget constraints**. In summary, this work has three main contributions.

1. We interpret LLM reasoning through the lens of an abstract reasoning tree, where the problem of early exiting is equivalent to identifying when this tree stops growing.
2. This view allows us to calibrate the decision rule for actively terminating generation.
3. Based on multiple language models and reasoning benchmarks, we provide empirical evidence that thought calibration is effective for efficient test-time scaling.

2 Background

2.1 Test-time scaling

Efficient inference Since current reasoning models are post-trained through reinforcement learning (Guo et al., 2025a), a number of works address the overthinking problem (Sui et al., 2025) as part of the reinforcement learning process (Han et al., 2024; Arora and Zanette, 2025; Hou et al., 2025). Other works focus on the inference-time problem of predicting when a language model should stop generating (Yang et al., 2025; Zhang et al., 2025; Ma et al., 2025). This papers falls under the latter category, which on a whole, is compatible with methods that reduce a language model's verbosity during post-training. Finally, another option is to achieve efficiency in terms of model architecture. Some works dynamically adapt compute cost (Lei, 2021; Leviathan et al., 2023), while others employ only a subset of all modules during sampling (Kim and Cho, 2021; Liu et al., 2022; Schuster et al., 2022). While these strategies operate over the Transformer stack, rather than the generation sequence length, many high-level ideas are broadly applicable to early exiting in our situation.

Self-consistency Self-consistency has been widely used to provide a self-supervised form of confidence during the sampling process (Wang et al., 2022). These methods aim to improve the quality of generated samples, often in situations where multiple samples may be sequentially generated (Mitchell et al., 2022; Madaan et al., 2023; Shi et al., 2023; Weng et al., 2023; Guo et al., 2025b). Consistency can also provide feedback for reasoning-focused reinforcement learning (Wang et al., 2024b). Several recent works have observed that confidence scores can be probed and calibrated from internal representations, to prioritize reasoning trajectories for subsequent runs (Li et al., 2024; Huang et al., 2025; Xie et al., 2024) or for early exiting, similar to this work (Zhang et al., 2025). Our key departure is that we calibrate the *decision rule* to terminate generation, rather than the probabilistic outputs of a probe. This reflects the online setting, where a probe is used to actively guide generation, rather than to filter trajectories post-hoc.

2.2 Conformal prediction and risk control

Conformal prediction quantifies the uncertainty in machine learning models by generating set-valued predictions (Shafer and Vovk, 2008; Angelopoulos and Bates, 2021). These methods are distribution-free and valid under finite samples, which makes them particularly attractive in real-world applications. Specifically, for an input x, a candidate output space $\mathcal{Y}$, and a predetermined error level ϵ, conformal prediction tests each potential outcome $y \in \mathcal{Y}$ by evaluating the null hypothesis: "output y corresponds to input x." The final prediction set consists of the outputs y for which this null hypothesis fails to be rejected, where the test statistic is known as a nonconformity score. Split conformal prediction leverages a separate training set to learn this nonconformity score (Vovk et al., 2005; Papadopoulos, 2008). The true outcome is included with probability at least $1 - \epsilon$, with guarantees that are typically marginal over draws of the test set and an exchangeable calibration set.

In the context of language modeling, conformal prediction has been adapted to calibrate the factuality (Mohri and Hashimoto, 2024; Cherian et al., 2024), reasoning consistency (Rubin-Toles et al., 2025a), and quality of generations (Quach et al., 2024; Qiu and Miikkulainen, 2024). Here, x may represent an input sequence of text, while y may be a language model output. Of these works, Rubin-Toles et al. (2025a) also introduces the idea of reasoning as coherency over a graph structure, based on logical deducibility. However, this and other methods are primarily designed for post-processing text that has already been generated, and they rely on external language models as scoring functions. As a result, these approaches are not calibrated to be used as decision rules for iterative testing, and the latency required to compute nonconformity scores renders them unsuitable for early exiting.

More recently, the Learn then Test (LTT) framework (Angelopoulos et al., 2021) extends the ideas in conformal prediction to control the risk of arbitrary loss functions, with guarantees over draws of the calibration set. One application of LTT is to convert model outputs into a calibrated decision rule, by viewing hyperparameter selection (e.g. discretization thresholds) as multiple hypothesis testing. Our method and several works in early exiting are built atop the LTT framework. Quach et al. (2024) calibrates a language model's *sampling* of output sets, similar to this work. Their goal is to generate sufficient outputs y until certain admissibility criteria have been fulfilled, e.g. correctness and diversity of information. However, the sampling process in Quach et al. (2024) is interactive, in the sense that each step requires an external verifier, and text may be added or removed at any point. As a result, this strategy is unsuitable for providing online decisions about when to stop. Schuster et al. (2022) also leverages LTT to calibrate a stopping rule to exit from a Transformer stack. Their method operates on individual tokens, similar in spirit to applications like speculative decoding (Leviathan et al., 2023). Our focus is on large, coherent thoughts for reasoning, where token-level uncertainties are less informative.

3 Thought Calibration

Thought calibration aims to efficiently utilize test-time compute by identifying when a language model's reasoning has converged. Section 3.1 introduces the theoretical framework through which we approach this problem (Figure 1), and Section 3.2 builds upon these abstractions to calibrate an online stopping rule for generation. Sections 3.3 and 3.4 describe three strategies for implementing thought calibration in practice, from both the modeling and data perspectives (Figure 2).

3.1 Problem setting

Given an input $x \in \mathcal{X}$, a reasoning language model generates a series of thoughts $y \in \mathcal{Y}$, before synthesizing the final output $z \in \mathcal{Z}$. For example, x may represent a math question; y is a sequence of reasoning steps; and z is the model's attempt at solving the question (Figure 1A). Manipulating the budget allocated to generating y directly impacts the quality of z (Muennighoff et al., 2025), but as the length of y increases, so too does the cost of inference. Our goal is to identify the point at which growing y no longer improves z.

To formalize these ideas, we introduce the notion of an abstract *reasoning graph* G, where nodes represent thoughts and directed edges represent entailment relationships (MacCartney and Manning, 2014). This graph is rooted at x, the input question. Nodes can be *serialized* into textual descriptions, and different paraphrases of the same idea represent a single node. Where it is clear, we refer to the abstract node and its textual representation interchangeably.

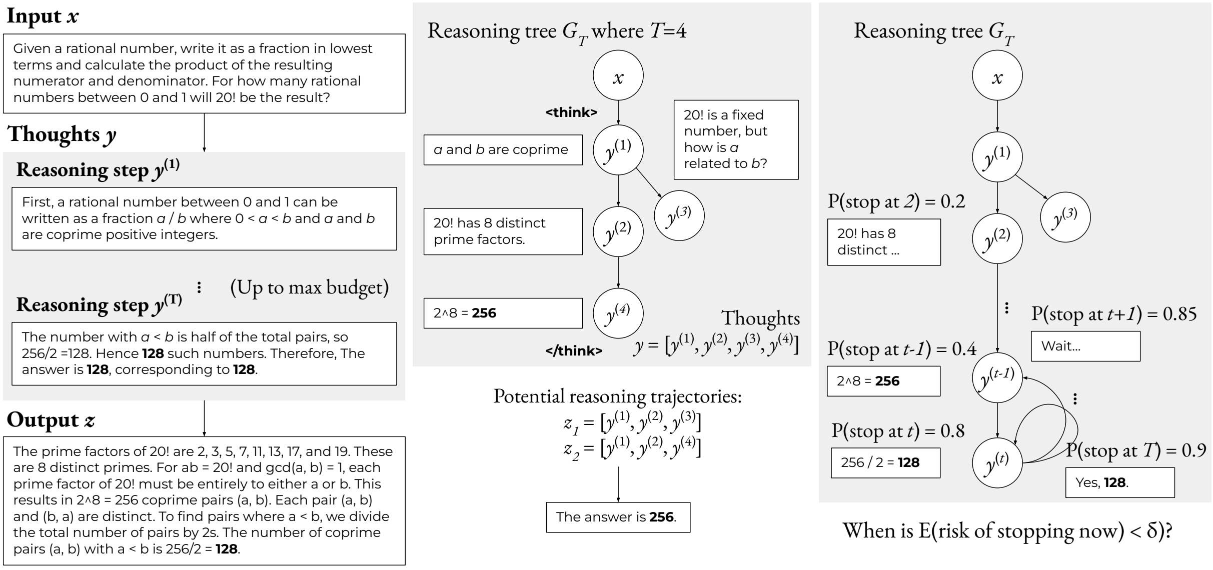
Definition 3.1. A *reasoning trajectory* z is a root-to-leaf walk in the reasoning graph G.

An arbitrary z need not be "complete" or "correct" with respect to the original question x. We use z^* to denote a walk that starts at x and ends at the right answer, which we assume to be incontrovertible. G uniquely determines the set of all root-to-leaf walks $\{z\}$, and thus, whether a language model has any chance of being correct in its final attempt.

Definition 3.2. A set of *thoughts* y is a walk, rooted at x, on the augmented graph G' in which every node is connected to each of its ancestors.

At each stage of sampling, a large language model either adds a leaf to G (novel thought), or takes one step in G' (backtracking or redundant generation). Let G_t be the reasoning graph at time t. If a language model terminates thinking at this point, it is expected to answer correctly if there exists a path in G_t that yields z^*. Thus, it would be ideal we could calibrate the language model such that with high probability,

$$\mathbb{P}\left(\mathbb{E}\left[\mathbb{1}[z^* \notin G_t] \leq \delta\right]\right) \geq 1 - \epsilon \qquad (1)$$



Given a rational number, write it as a fraction in lowest terms and calculate the product of the resulting numerator and denominator. For how many rational numbers between 0 and 1 will 20! be the result?

First, a rational number between 0 and 1 can be written as a fraction *a* / *b* where 0 < *a* < *b* and *a* and *b* are coprime positive integers.

The number with *a* < *b* is half of the total pairs, so 256/2 =128. Hence **128** such numbers. Therefore, The answer is **128**, corresponding to **128**.

The prime factors of 20! are 2, 3, 5, 7, 11, 13, 17, and 19. These are 8 distinct primes. For ab = 20! and gcd(a, b) = 1, each prime factor of 20! must be entirely to either a or b. This results in 2^8 = 256 coprime pairs (a, b). Each pair (a, b) and (b, a) are distinct. To find pairs where a < b, we divide the total number of pairs by 2s. The number of coprime pairs (a, b) with a < b is 256/2 = **128**.

(A) Reasoning language models scale at test time through longer chains of thought, at significant compute cost.

(B) Terminating too early hurts model performance due to under exploration.

(C) Our goal is to confidently decide when y_t will be consistent with y_T, based on when the reasoning graph stops changing.

Figure 1: Overview of the problem and our goal. Illustrated example based on s1K-1.1 (Muennighoff et al., 2025).

for some risk tolerance δ and error level $\epsilon \in (0, 1)$. In principle, a language model could enumerate the space of graphs in a combinatorial search. However, it is far from guaranteed that this graph can be tractably found. Instead, we focus on the consistency between reasoning graphs.

Definition 3.3. Thoughts y and y' are *consistent* if they can be represented by the reasoning graph G.

In particular, if a language model repeatedly revisits a step to arrive at the same conclusion, or traverses the same ideas in a different order, the resultant graph does not change (Figure 1C). Let $y_t := [y^{(i)} \dots y^{(t)}]$ and G_t be the in-progress thoughts and reasoning graph after t steps, and let T be the maximum inference budget (token or model limit). Instead of enforcing that G_t contains z^*, it is more reasonable to guarantee that

$$\mathbb{P}\left(\mathbb{E}\left[\mathbb{1}[G_t \neq G_T] \leq \delta\right]\right) \geq 1 - \epsilon. \tag{2}$$

Due to the sequential nature of generation, G_t is always a (not necessarily strict) subset of G_T.

3.2 Calibrating the stopping rule

Suppose we have a calibration dataset $\mathcal{D}_{\text{cal}}$, which contains exchangeable points $\{(x_i, y_i)\}_{i=1}^n$. Given a new example x, let y_t denote the language model's thoughts after t sampling steps, and let y_T denote the maximum set of thoughts. Our goal is to find the smallest t that fulfills Equations 1 or 2, based on the distribution of $\mathcal{D}_{\text{cal}}$. During the sampling process, however, we do not know z^* or G_T, so we must estimate the quantities inside the expectation using a surrogate function f. Here, $\mathcal{D}_{\text{cal}}$ serves to calibrate f such that

$$\mathbb{P}\left(\mathbb{E}\left[R(y_t) \leq \delta \mid \mathcal{D}_{\text{cal}}\right]\right) \geq 1 - \epsilon \tag{3}$$

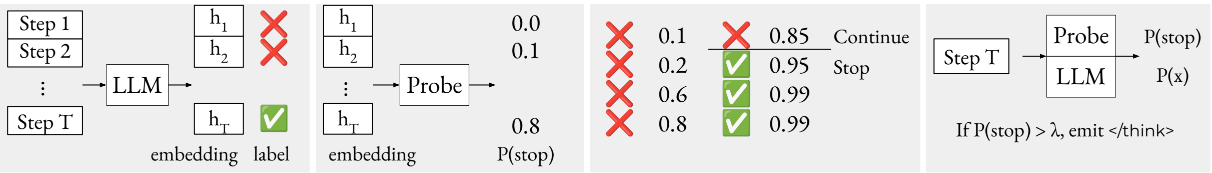
where R is a bounded risk function associated with f. For example, f may be a linear probe on the hidden representations of thought steps $y^{(i)}$, and its output may be a binary prediction. A potential decision rule could take the form of a threshold λ, where if $f(y^{(t)}) \geq \lambda$, we terminate thinking.

Similar to Schuster et al. (2022) and Quach et al. (2024), we follow the Learn then Test framework to select a valid set of λs that provide our desired guarantees (Angelopoulos et al., 2021). On a high level, hyperparameter selection is viewed as a multiple hypothesis testing problem. Let Λ be a finite set of configurations, where each $\lambda_j \in \Lambda$ is associated with the null hypothesis,

$$H_j : \mathbb{E}[R(y_t) > \delta]. \tag{4}$$

The set of valid $\Lambda_{\text{valid}} \subseteq \Lambda$ is the set of λ_j for which we *fail to reject* H_j. In particular, selecting the earliest stopping time is equivalent to identifying the smallest $\lambda \in \Lambda_{\text{valid}}$.

Theorem 3.4 (Adapted from theorem 1 in (Angelopoulos et al., 2021))**.** *Suppose p_j is super-uniform under H_j for all j. Let $\mathcal{A}$ be a family-wise error rate (FWER) controlling algorithm at level ϵ. Then $\Lambda_{\text{valid}} = \mathcal{A}(p_1, \dots, p_m)$ satisfies Equation 3.*



1. Sample reasoning trajectory. At each step: save last layer embeddings, force LLM to answer, check consistency with final answer.

2. Train probe to predict consistency with final answer, based on step embeddings.

3. Calibrate probe on exchangeable dataset via *learn then test*, yielding thresholds for predefined error levels.

4. At inference time, run probe as additional head. Stop based on calibrated thresholds.

Figure 2: High level overview of the algorithm.

Theorem 3.4 specifies that any FWER-controlling algorithm $\mathcal{A}$ can be used with an appropriate p-value to identify Λ_{valid}. Angelopoulos et al. (2021) describes several algorithms to search over Λ, and we follow the fixed sequence testing method, since in principle, our risks are expected to be monotonic ($G_t \subseteq G_T$).

Specifically, let $\Lambda = \{\lambda_1, \dots, \lambda_m\}$ be a descending grid of parameters. Intuitively, larger λ correspond to more permissive thresholds, e.g. allowing a language model to generate for longer.

1. For each j, we compute a valid p-value p_j, e.g. the binomial tail bound p-value, following (Quach et al., 2024):

$$p_\lambda^{BT} := \mathbb{P}(\mathrm{Binom}(n, \epsilon) \leq n\hat{R}_n(\lambda)). \quad (5)$$

2. If $p_j \leq \epsilon$, we reject H_j and continue. Otherwise, we return λ_{j-1} as the smallest valid threshold for error rate ϵ.

This process yields the binarization threshold for f, where we stop generating when $f(y_t) \geq \lambda_{j-1}$.

3.3 Estimating empirical risk

On a high level, the surrogate function f should reflect the consistency of y_t with expected future generations. Ideally, we would be able to access the graphical structure of G_t, as any repetitions or redundant walks in y_t would be immediately evident. However, since autoregressive language models generate left-to-right, without explicitly conforming to any higher-level structure, we cannot operate directly over G. Instead, we introduce three approaches for designing f in practice.

We first briefly consider the simple case suggested by Equation 1: if we terminate thinking now, is the language model able to answer correctly? That is, we could define

$$f_{\text{correct}}(y_t) := \mathbb{P}(\text{LLM is correct based on } y_t) \quad (6)$$

$$R_{\text{correct}}(y_t) := \mathbb{1}\{\text{LLM is correct}\} \cdot (1 - f_{\text{correct}}(y_t)) + \mathbb{1}\{\text{LLM is wrong}\} \cdot f_{\text{correct}}(y_t). \quad (7)$$

However, there are several drawbacks of this implementation. By construction, the calibration dataset only contain questions that can eventually be answered, which is not true in general. Though the space of graphs is countable, it is unlikely that a language model can efficiently explore the entire space. In other words, the language model may realistically *never* answer correctly. Thus, setting $\lambda = 1$ is *not* guaranteed to be risk controlling. With this definition of $R_{\text{correct}}(y)$, calibrating based on correctness also requires supervised labels. While this is not an issue on standard benchmarks, it is harder to obtain labels (user feedback) in practice.

To address these challenges, we introduce two additional strategies for estimating graph consistency. First, a language model's final attempt z can be viewed as a distillation of its overall reasoning structure. Thus, we compare the language model's attempt z_t after t steps, to the eventual attempt z_T at the maximum reasoning budget. This yields

$$f_{\text{consistent}}(y_t) := \mathbb{P}(z_t \text{ is the same as } z_T) \quad (8)$$

$$R_{\text{consistent}}(y_t) := \mathbb{1}\{\text{consistent}\} \cdot (1 - f_{\text{consistent}}(y_t)) + \mathbb{1}\{\text{inconsistent}\} \cdot f_{\text{consistent}}(y_t) \quad (9)$$

These values can be determined even for intractable problems, as long as the extended reasoning produces no new insights, and does not require labels of correctness.

Finally, any particular z only represents a single walk through G. Due to stochasticity, two differing attempts could be sampled from the same graph, which is no longer changing. Towards this end, we observed that language models often reiterate redundant information, after having reached the correct answer or the extent of its abilities. Probing for novelty should suffice to capture this phenomena. In practice, however, we found that the following formulation was easier for our verifier to implement, as checking for novelty involves long context reasoning over all previous thoughts, which can be challenging (Wang et al., 2024a).

$$f_{\text{novel leaf}}(y_t) := \mathbb{P}(y^{(t)}\text{is leaf}) \cdot (1 - \mathbb{P}(y^{(t)}\text{is novel})) \quad (10)$$

$$R_{\text{novel leaf}}(y_t) := \mathbb{1}\{\text{LLM inconsistent}\} \cdot f_{\text{novel leaf}}(y_t) + \mathbb{1}\{\text{LLM consistent}\} \cdot (1 - f_{\text{novel leaf}}(y_t)). \quad (11)$$

We reuse the labels for consistency due to ease of verification compared to novelty.

3.4 Implementation details

Overview Given any language model and a set of relevant prompts, we first sample maximum-budget reasoning trajectories. These trajectories are truncated at intermediate reasoning steps, and we force the same model to produce a final attempt at each step. We train lightweight probes to predict whether the intermediate answers are consistent with the original, maximum-budget answer. These probes are calibrated on a separate, exchangeable calibration set: given a predefined error level ϵ, the procedure returns a corresponding probe threshold λ, which is used for the test-time decision rule. Figure 2 illustrates the overall workflow.

Reasoning steps To separate a reasoning trajectory y into individual steps $\{y^{(i)}\}$, we use sections delimited by `\n\n`, which also contain `wait` or `but`. We observed that individual tokens representations vary significantly. Thus, each step uses the mean last-layer representation of its tokens, followed by dimensionality reduction via PCA to $d = 256$.

Probes To estimate each of quantities in Equations (6) to (11), we train linear probes on these step-level representations. The final probabilities are averaged over a window of 10 steps for smoothness, before calibration. For evaluation, we use a grid of ϵ ranging from 0.05 to 0.5, with precise thresholds selected to roughly match the token range of baselines. During development, we experimented with more complex architectures, e.g. Transformer to predict leaves as a sequence labeling task (Appendix B.1). However, to avoid overfitting on our limited training set, we chose to focus on simple and efficient linear probes. Concurrent work (Zhang et al., 2025) also finds that model confidence can often be extracted linearly. In our experiments, we use three reasoning models: DeepSeek-R1 distilled Qwen 2.5 32B and Llama 3.3 70B (Guo et al., 2025a; Grattafiori et al., 2024; Yang et al., 2024), and QwQ 32B (Team, 2025).

The ground truth labels for these probes are obtained by prompting a separate language model (Qwen 3 32B). **Correct**: We truncate thinking trajectories to desired lengths, append the `<\think>` token, and prompt the language model for the final answer, which is compared to the ground truth (Muennighoff et al., 2025). **Consistent:** The same outputs can be used to check whether G_t is consistent with G_T, by comparing intermediate attempts z_t to maximum budget attempt z_T. **Leaf:** We annotate whether each step $y^{(i)}$ is a leaf in G by asking a separate language model to identify whether it makes an attempt to answer the original question x, regardless of correctness. **Novel:** We provide a separate language model with all previous thoughts $y^{(1)} \dots y^{(i-1)}$ and ask whether the new step $y^{(i)}$ provides additional information. All prompts can be found in Appendix A and were run on 4 A6000 GPUs using vLLM (Kwon et al., 2023) and lmdeploy (Contributors, 2023).

Evaluation We evaluate the correctness of all final attempts using the GPT 4.1 API, between April 15, 2025 and May 15, 2025. For datasets with no ambiguity (multiple choice, numeric answers), we trimmed the final attempts to 200 characters, to prevent the LLM from "cheating" by using additional thinking budget after the `</think>` token.

4 Experiments

4.1 Settings

Datasets Our experiments focus on efficient language model reasoning across tasks which vary in content, format, and difficulty. In particular, we leverage the following datasets.

s1K-1.1 (Muennighoff et al., 2025) is a curated *training set* for distilling reasoning abilities through data. This dataset contains 1000 difficult math and science questions, along with thought trajectories generated by DeepSeek-R1 (Guo et al., 2025a). As a proof of concept, we split the s1K-1.1 dataset into training, testing, and calibration (500, 50, 450, in dataset order). We use the training set to develop our probes, which are calibrated on the calibration set and evaluated on the testing set.

We also consider three common reasoning benchmarks solely for testing. **AIME-24** is the 2024 iteration of the American Invitational Mathematics Examination.[2] This dataset contains math questions whose answers take on integers be-

---

[2] https://maa.org/maa-invitational-competitions/

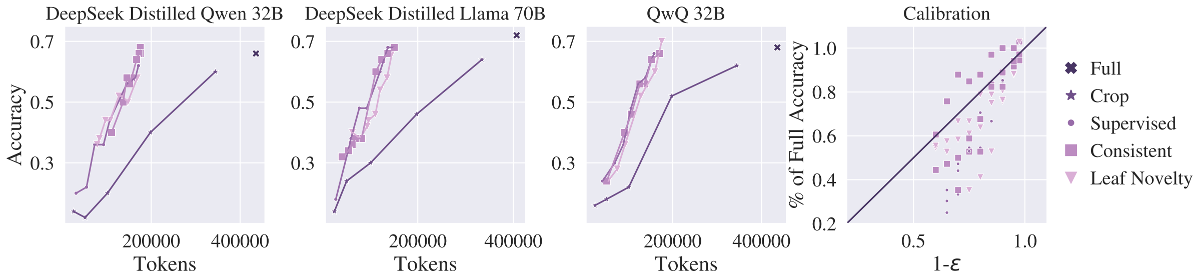


Figure 3: On in-distribution data (held-out test split on s1K), variants of thought calibration achieve up to a 60% reduction in thinking tokens while maintaining full performance. Top right point: Complete DeepSeek-R1 thought trajectory from (Muennighoff et al., 2025). Crop: Fix thinking budget at 512, 1024, 2048, 4096, and 8192 tokens. Supervised: exit based on predicted likelihood of correctness. Consistent, and Leaf Novelty: exit based on predicted consistency of answer or graph. Supervised is over confident, since the test set contains unsolvable problems.

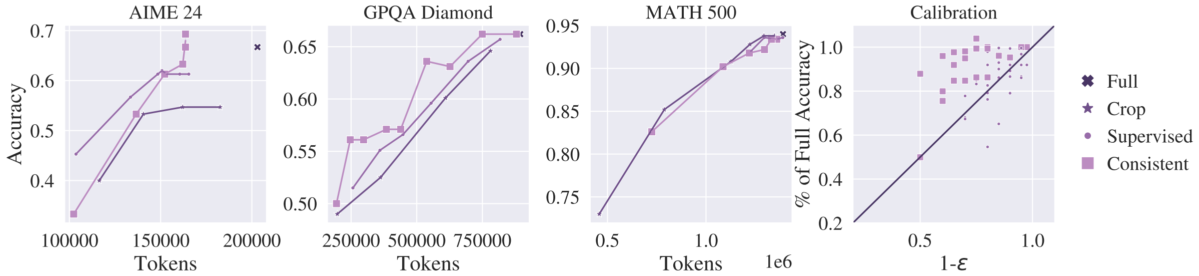


Figure 4: We applied thought calibration probes for DeepSeek-distilled Qwen-2.5 32B on standard math and science benchmarks, which may be out-of-distribution compared to the training and calibration sets, drawn from s1K. We achieve up to a 20% reduction in thinking tokens. While Consistent generally remains below the predetermined error rates, Supervised is overconfident (as expected).

tween 0 and 999. **GPQA Diamond** (Rein et al., 2024) is a PhD-level math and science reasoning benchmark with multiple choice answers. **MATH 500** (Hendrycks et al., 2021; Lightman et al., 2023) is a curated subset of the MATH dataset, which competition math questions of various levels. Note that while s1K-1.1 contains examples of both mathematical and scientific questions, the format and subsequent reasoning patterns may vary. For example, while s1K-1.1 is open-ended, the various choices in GPQA must be compared. Thus, we view these three datasets as "out of distribution" from s1K-1.1, which is itself diverse.

Models We evaluate the three variants of thought calibration: the supervised probe for correctness (Equation 6, **Supervised**); the consistency probe (Equation 8, **Consistent**); and the lack of novelty probe (Equation 10, **Novel Leaf**). To contextualize our experimental results, we also consider a naive budget-forcing baseline (**Crop**). Specifically, we set a fixed token budget for thinking (ranging from 1024 to the full trajectory). Once the language model reaches this budget, thinking is immediately terminated and the model is prompted for a final answer. This reflects both the practical use case of setting a limit on maximum generation tokens, and the strategy employed by Muennighoff et al. (2025). Finally, concurrent work has also observed that probes for correctness (Zhang et al., 2025) are effective for early exiting. While this design may not be valid for risk control in practice (LLMs are not guaranteed to ever answer correctly), the **Supervised** baseline is similar to this work.

4.2 In-distribution setting

We start with the case where we have access to samples x that are drawn from the same distribution as our eventual application. For example, a model provider may possess typical examples of user data. Our goals are to lower the overall test-time budget while maintaining accuracy, and to control any necessary drops in performance based on our predetermined error levels. In Figure 3, we observe that these probes are able to **reduce the number of thinking tokens by over half for all three models**, with minimal impact to overall performance. With respect to calibration, the Supervised probe is quite poorly calibrated, especially at lower values of ϵ. All other probes are well calibrated at $\epsilon < 0.1$, though variance is higher outside of this range. This may be due to distribution shift, resulting from the small test split (to maximize training and calibration data for subsequent evaluations).

4.3 Generalization setting

Next, we consider the case in which the data we have is related, but *not* drawn from the same distribution as our eventual application. To emulate this setting, we apply the supervised and consistent Qwen 32B probes, developed on the s1K-1.1 dataset, to common reasoning benchmarks (Rein et al., 2024; Lightman et al., 2023). Overall, we are able to improve (AIME 24, GPQA) or match (MATH 500) the efficiency of the budget forcing baseline – even achieving slight **gains in performance on AIME 24**, perhaps by trimming distracting thoughts (Figure 4). Notably, even though the Supervised probe had access to more information (ground truth answers), **the Consistent probe consistently generalizes better, both in terms of efficiency and calibration**. Here, the Consistent probe fulfills the theoretical guarantees, while the Supervised probe remains over-confident.

4.4 Additional analysis

Figure 5 illustrates that thought calibration probes prioritizes the termination of problems which cannot be solved, even at full budget – perhaps hinting that the language model may have been stuck in a cycle of reasoning, without novel progress. Compared to the naive cropping strategy, thought calibration's input-dependent decision also demonstrate significant variance in the amount of tokens across different problems.

We also examine a specific instance from our s1K-1.1 testing split in Figure 6 (s1K is a distillation dataset, so this diagram does not leak real test examples). The language model reaches the correct answer after 38 steps (out of 48 steps). As the model backtracks, the predicted consistency (with the expected final answer) drops; and as the model returns to the answer, confidence increases, higher than before. This reaffirms that self-consistency is indeed a powerful indication of correctness, both distilled into a predictive model, and over the course of sampling.

5 Discussion

Since inference costs scale directly with model usage, reasoning efficiency has become a critical aspect of deploying language models in practice. This paper introduced thought calibration, a framework that enables efficient and controllable inference-time scaling. Thought calibration produces lightweight probes, designed to be run during generation, that identify when a language model has reached a consistent answer. As a decoding strategy, this work is complementary to methods that aim to train more efficient models. In addition, the probes described here can be used as classifiers, to identify examples of efficient reasoning for further distillation. Finally, while this paper only addresses the problem of exiting early from reasoning, the broader question of how to calibrate the steering of reasoning models remains unanswered, and is an interesting area for further research.

6 Limitations

There are several limitations of our work. Since our method is built atop the Learn then Test framework (Angelopoulos et al., 2021), our theoretical guarantees are only valid over draws of the calibration set. In practice, this means that the calibration data must be sufficiently similar to the actual application. Furthermore, due to our small training and calibration datasets, we implement our framework primarily through linear probes. In Appendix B.1, we found that more complex architectures may lead to slightly better performance in some cases, and the gap is expected to be larger if more training data can be gathered. We leave further investigations regarding the probe architecture to future work.

Acknowledgements

This material is based upon work supported by the National Science Foundation Graduate Research Fellowship under Grant No. 1745302. We would like to acknowledge support from the Machine Learning for Pharmaceutical Discovery and Synthesis (MLPDS) consortium, and the Abdul Latif Jameel Clinic for Machine Learning in Health.

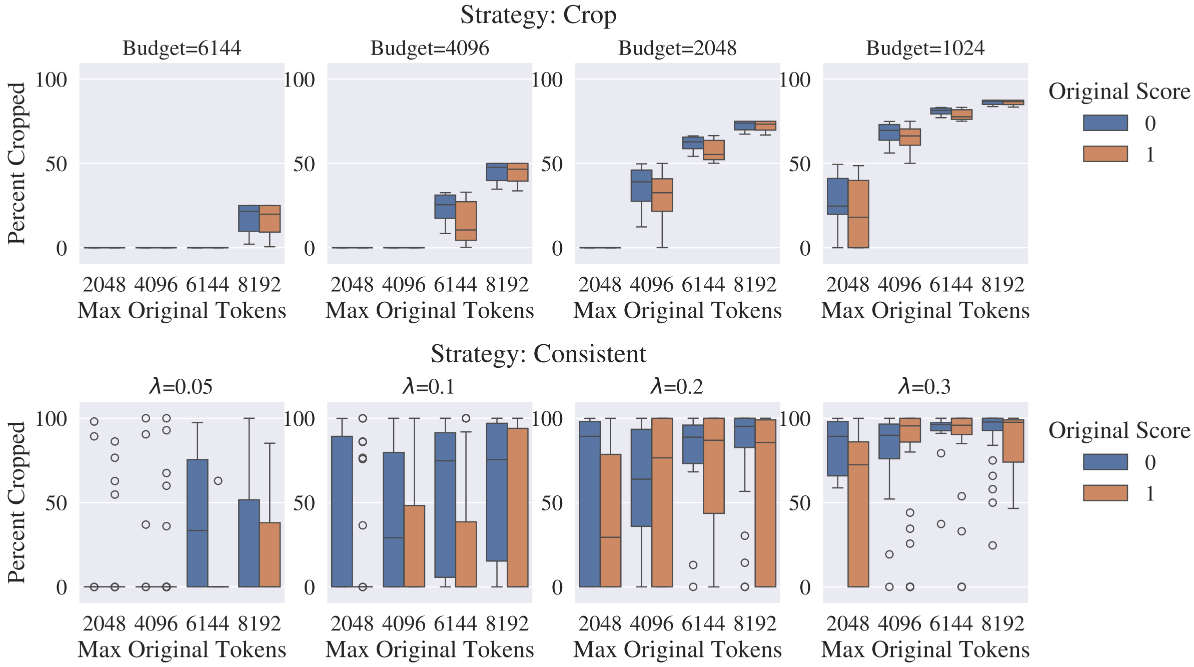
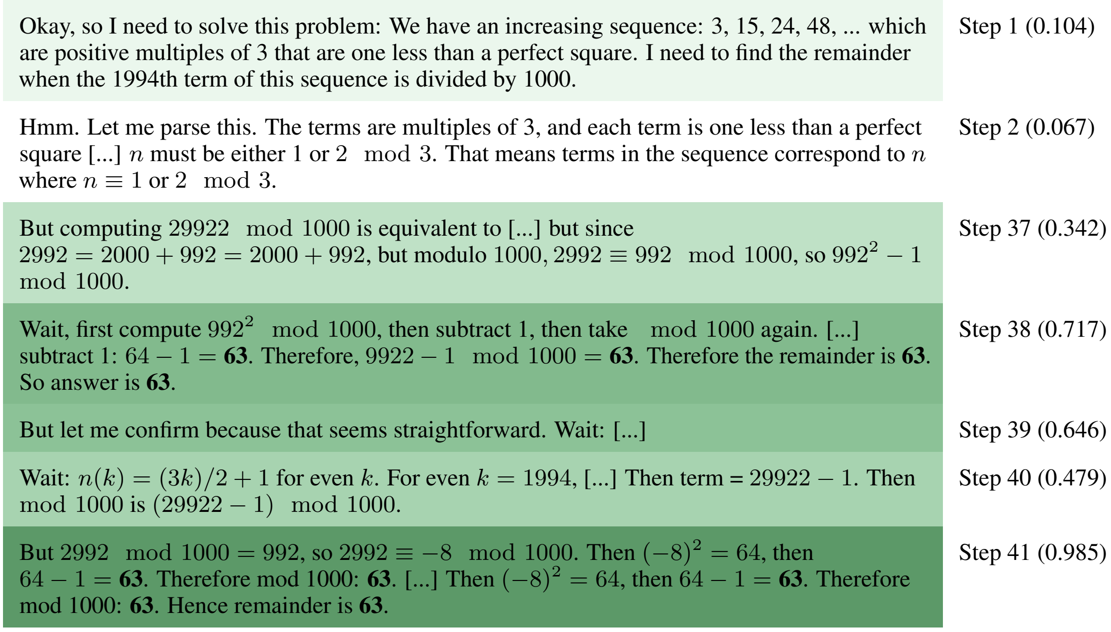


Figure 5: Proportion of prompt tokens removed, for different thresholds, stratified by full thought length and whether the original model was able to solve the problem. Top: Naive max token thresholding. Bottom: Consistency calibration, DeepSeek-R1 distilled Qwen 32B, over GPQA Diamond. Cropping reduces token lengths uniformly, regardless of the input characteristics. Thought calibration has a preference for first trimming longer thoughts and cases where the language model tries but fails to make progress.



Okay, so I need to solve this problem: We have an increasing sequence: 3, 15, 24, 48, ... which are positive multiples of 3 that are one less than a perfect square. I need to find the remainder when the 1994th term of this sequence is divided by 1000.	Step 1 (0.104)
Hmm. Let me parse this. The terms are multiples of 3, and each term is one less than a perfect square [...] n must be either 1 or 2 $\mod 3$. That means terms in the sequence correspond to n where $n \equiv 1$ or $2 \mod 3$.	Step 2 (0.067)
But computing $29922 \mod 1000$ is equivalent to [...] but since $2992 = 2000 + 992 = 2000 + 992$, but modulo 1000, $2992 \equiv 992 \mod 1000$, so $992^2 - 1$ $\mod 1000$.	Step 37 (0.342)
Wait, first compute $992^2 \mod 1000$, then subtract 1, then take $\mod 1000$ again. [...] subtract 1: $64 - 1 = \mathbf{63}$. Therefore, $9922 - 1 \mod 1000 = \mathbf{63}$. Therefore the remainder is **63**. So answer is **63**.	Step 38 (0.717)
But let me confirm because that seems straightforward. Wait: [...]	Step 39 (0.646)
Wait: $n(k) = (3k)/2 + 1$ for even k. For even $k = 1994$, [...] Then term = $29922 - 1$. Then $\mod 1000$ is $(29922 - 1) \mod 1000$.	Step 40 (0.479)
But $2992 \mod 1000 = 992$, so $2992 \equiv -8 \mod 1000$. Then $(-8)^2 = 64$, then $64 - 1 = \mathbf{63}$. Therefore mod 1000: **63**. [...] Then $(-8)^2 = 64$, then $64 - 1 = \mathbf{63}$. Therefore mod 1000: **63**. Hence remainder is **63**.	Step 41 (0.985)

Figure 6: DeepSeek-R1 distilled Llama 70B Consistency probe on s1K-1.1 example from our test split, where color intensity is proportional to $\mathbb{P}(\text{consistent})$. The language model first reaches the correct answer in Step 38, backtracks with lower confidence, and returns to the answer in Step 41.

References

Anastasios N Angelopoulos and Stephen Bates. 2021. A gentle introduction to conformal prediction and distribution-free uncertainty quantification. *arXiv preprint arXiv:2107.07511*.

Anastasios N Angelopoulos, Stephen Bates, Emmanuel J Candès, Michael I Jordan, and Lihua Lei. 2021. Learn then Test: Calibrating predictive algorithms to achieve risk control. *arXiv preprint arXiv:2110.01052*.

Daman Arora and Andrea Zanette. 2025. Training language models to reason efficiently. *arXiv preprint arXiv:2502.04463*.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, and 1 others. 2024. Do not think that much for 2+ 3=? on the overthinking of o1-like LLMs. *arXiv preprint arXiv:2412.21187*.

John Cherian, Isaac Gibbs, and Emmanuel Candes. 2024. Large language model validity via enhanced conformal prediction methods. *Advances in Neural Information Processing Systems*, 37:114812–114842.

LMDeploy Contributors. 2023. Lmdeploy: A toolkit for compressing, deploying, and serving llm. `https://github.com/InternLM/lmdeploy`.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025a. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Jiacheng Guo, Yue Wu, Jiahao Qiu, Kaixuan Huang, Xinzhe Juan, Ling Yang, and Mengdi Wang. 2025b. Temporal consistency for llm reasoning process error identification. *arXiv preprint arXiv:2503.14495*.

Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. 2024. Token-budget-aware LLM reasoning. *arXiv preprint arXiv:2412.18547*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Bairu Hou, Yang Zhang, Jiabao Ji, Yujian Liu, Kaizhi Qian, Jacob Andreas, and Shiyu Chang. 2025. ThinkPrune: Pruning long chain-of-thought of LLMs via reinforcement learning. *arXiv preprint arXiv:2504.01296*.

Chengsong Huang, Langlin Huang, Jixuan Leng, Jiacheng Liu, and Jiaxin Huang. 2025. Efficient test-time scaling via self-calibration. *arXiv preprint arXiv:2503.00031*.

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

Gyuwan Kim and Kyunghyun Cho. 2021. Length-adaptive transformer: Train once with length drop, use anytime with search. In *Joint Conference of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL-IJCNLP 2021*, pages 6501–6511. Association for Computational Linguistics (ACL).

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

Tao Lei. 2021. When attention meets fast recurrence: Training language models with reduced compute. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pages 19274–19286. PMLR.

Yiwei Li, Peiwen Yuan, Shaoxiong Feng, Boyuan Pan, Xinglin Wang, Bin Sun, Heda Wang, and Kan Li. 2024. Escape sky-high cost: Early-stopping self-consistency for multi-step reasoning. In *The Twelfth International Conference on Learning Representations*.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*.

Zhuang Liu, Zhiqiu Xu, Hung-Ju Wang, Trevor Darrell, and Evan Shelhamer. 2022. Anytime dense prediction with confidence adaptivity. In *International Conference on Learning Representations*.

Wenjie Ma, Jingxuan He, Charlie Snell, Tyler Griggs, Sewon Min, and Matei Zaharia. 2025. Reasoning models can be effective without thinking. *arXiv preprint arXiv:2504.09858*.

Bill MacCartney and Christopher D Manning. 2014. Natural logic and natural language inference. In *Computing Meaning: Volume 4*, pages 129–147. Springer.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, and 1 others. 2023. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36:46534–46594.

Eric Mitchell, Joseph Noh, Siyan Li, Will Armstrong, Ananth Agarwal, Patrick Liu, Chelsea Finn, and Christopher D Manning. 2022. Enhancing self-consistency and performance of pre-trained language models through natural language inference. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 1754–1768.

Christopher Mohri and Tatsunori Hashimoto. 2024. Language models with conformal factuality guarantees. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org.

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

Harris Papadopoulos. 2008. Inductive conformal prediction: Theory and application to neural networks. In *Tools in artificial intelligence*. Citeseer.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Xin Qiu and Risto Miikkulainen. 2024. Semantic density: Uncertainty quantification for large language models through confidence measurement in semantic space. In *Advances in Neural Information Processing Systems*, volume 37, pages 134507–134533.

Victor Quach, Adam Fisch, Tal Schuster, Adam Yala, Jae Ho Sohn, Tommi S Jaakkola, and Regina Barzilay. 2024. Conformal language modeling. In *The Twelfth International Conference on Learning Representations*.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. 2024. GPQA: A graduate-level Google-proof Q&A benchmark. In *First Conference on Language Modeling*.

Maxon Rubin-Toles, Maya Gambhir, Keshav Ramji, Aaron Roth, and Surbhi Goel. 2025a. Conformal language model reasoning with coherent factuality. In *The Thirteenth International Conference on Learning Representations*.

Maxon Rubin-Toles, Maya Gambhir, Keshav Ramji, Aaron Roth, and Surbhi Goel. 2025b. Conformal language model reasoning with coherent factuality. In *The Thirteenth International Conference on Learning Representations*.

Tal Schuster, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Tran, Yi Tay, and Donald Metzler. 2022. Confident adaptive language modeling. *Advances in Neural Information Processing Systems*, 35:17456–17472.

Glenn Shafer and Vladimir Vovk. 2008. A tutorial on conformal prediction. *Journal of Machine Learning Research*, 9(3).

Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed H Chi, Nathanael Schärli, and Denny Zhou. 2023. Large language models can be easily distracted by irrelevant context. In *International Conference on Machine Learning*, pages 31210–31227. PMLR.

Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Hanjie Chen, and 1 others. 2025. Stop overthinking: A survey on efficient reasoning for large language models. *arXiv preprint arXiv:2503.16419*.

Qwen Team. 2025. QwQ-32B: Embracing the power of reinforcement learning.

Vladimir Vovk, Alexander Gammerman, and Glenn Shafer. 2005. *Algorithmic learning in a random world*, volume 29. Springer.

Minzheng Wang, Longze Chen, Cheng Fu, Shengyi Liao, Xinghua Zhang, Bingli Wu, Haiyang Yu, Nan Xu, Lei Zhang, Run Luo, and 1 others. 2024a. Leave no document behind: Benchmarking long-context LLMs with extended multi-doc QA. *CoRR*.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. 2024b. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9426–9439.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Yixuan Weng, Minjun Zhu, Fei Xia, Bin Li, Shizhu He, Shengping Liu, Bin Sun, Kang Liu, and Jun Zhao. 2023. Large language models are better reasoners with self-verification. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 2550–2575.

Zhihui Xie, Jizhou Guo, Tong Yu, and Shuai Li. 2024. Calibrating reasoning in language models with internal consistency. *arXiv preprint arXiv:2405.18711*.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, and 1 others. 2024. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.

Chenxu Yang, Qingyi Si, Yongjie Duan, Zheliang Zhu, Chenyu Zhu, Zheng Lin, Li Cao, and Weiping Wang. 2025. Dynamic early exit in reasoning models. *arXiv preprint arXiv:2504.15895*.

Anqi Zhang, Yulin Chen, Jane Pan, Chen Zhao, Aurojit Panda, Jinyang Li, and He He. 2025. Reasoning models know when they're right: Probing hidden states for self-verification. *arXiv preprint arXiv:2504.05419*.

Dan Zhang, Sining Zhoubian, Ziniu Hu, Yisong Yue, Yuxiao Dong, and Jie Tang. 2024a. Rest-mcts*: Llm self-training via process reward guided tree search. *Advances in Neural Information Processing Systems*, 37:64735–64772.

Di Zhang, Xiaoshui Huang, Dongzhan Zhou, Yuqiang Li, and Wanli Ouyang. 2024b. Accessing gpt-4 level mathematical olympiad solutions via monte carlo tree self-refine with llama-3 8b. *arXiv preprint arXiv:2406.07394*.

A Prompts

The following prompt was used to force the model (DeepSeek-R1 distilled Qwen 32B and Llama 70B, QwQ 32B) to produce an answer after a fixed number of thinking steps. Following the recommendation of Guo et al. (2025a) and Team (2025), we do not include a system prompt. We apply the chat template to user prompt before concatenating the "in-progress" thoughts. Adapted from (Muennighoff et al., 2025).

```
<bos><User>
{question}
Please reason step by step, and put your final answer within \\boxed{{}}.
<Assistant>
<think>
{thoughts}
</think>
Final Answer:
```

The following prompt was used to obtain labels for $\mathbb{P}(\text{correct})$ (Equation 6) using Qwen 3 32B. This prompt was also used to evaluate answers using GPT 4.1. Adapted from (Muennighoff et al., 2025).

```
You are an AI assistant for grading a science problem. The user will provide you with the question itself, the correct answer, and the student's attempt. Your job is to judge whether the attempt is correct by comparing it with the correct answer. If the correct answer is a number or choice, there should be no ambiguity, and you should directly compare the answer and the final result. If the attempt is incomplete, you should mark it as wrong. If the correct answer involves going through the entire reasoning process, you should judge the result based on whether the reasoning process is correct, compared to correct answer.
Do NOT try to solve the problem yourself. Only grade the attempt based on the correct answer.
The user will provide the attempt and the correct answer in the following format:
# Problem
{problem}
## Correct answer
{solution}
## Student attempt
{attempt}
Explain your reasoning concisely, and end your response on a new line with only "Yes" or "No" (without quotes).
```

The following prompt was used to obtain labels for $\mathbb{P}(\text{consistent})$ (Equation 8) using Qwen 3 32B.

```
You are an AI assistant for grading a science problem. The user will provide you with the question itself and two student attempts. Your job is to judge whether the two students arrive at the same answer. If question asks for a single numerical answer, there should be no ambiguity, and you should directly compare the two answers. If the question asks for multiple parts, the two attempts are identical if only if all of the parts arrive at the same conclusion.
Do NOT try to solve the problem yourself. Only grade whether the two attempts are the same.
The user will provide the problem and two attempts in the following format:
# Problem
{problem}
## Attempt 1
{attempt1}
## Attempt 2
{attempt2}
Explain your reasoning concisely, and end your response on a new line with only "Yes" or "No" (without quotes).
```

The following prompt was used to obtain labels for $\mathbb{P}(\text{leaf})$ (Equation 10) using Qwen 3 32B.

```
You are an AI assistant for parsing LLM outputs. The user will provide you with the question and an intermediate reasoning step. Your job is to judge whether the given step contains an attempt at a final answer.
Do NOT attempt to solve the problem yourself. It does not matter if the answer is correct. Only comment on whether an attempt has been made.
The user will provide the problem and reasoning steps in the following format:
# Problem
{problem}
# Reasoning step
{reasoning step}
Explain your reasoning, and end your response on a new line with only "Yes" or "No" indicating whether or the given step makes an attempt at providing the final answer.
```

The following prompt was used to obtain labels for $\mathbb{P}(\text{novel})$ (Equation 10) using Qwen 3 32B.

```
You are an AI assistant for assessing the quality of logical reasoning. The user will provide you with the question and an incomplete attempt, consisting of a series of reasoning steps. Your job is to judge whether current step appears to provide additional information, compared to the previous steps. If the current step is correct and novel, it is useful. If the current step is wrong or redundant, then it is not useful.
Do NOT try to solve the problem yourself. It does not matter if the attempt is not complete. Only comment on whether the current step is useful.
The user will provide the problem and reasoning steps in the following format:
# Problem
```

```
{problem}
# Reasoning
## step 1
{reasoning step 1}
## step 2
{reasoning step 2}
...
## step k
{reasoning step k}
...
## current step
{current reasoning step}
Explain your reasoning, and end your response on a new line with only "Yes" if the current step provides new information or "No"
otherwise (without quotes).
```

B Implementation details

B.1 Design and implementation of model probes

We tried several architectures, before deciding upon linear probes for simplicity and to avoid overfitting. The differences in performance are not always consistent and the generalization gap is quite large (Table 1). Since our main focus is on calibration, and it requires significant compute to produce and evaluate scaling curves, we consider more exhaustive exploration of alternate architectures as future work.

MLP The input is a single representation $h^{(t)}$ corresponding to single reasoning step $y^{(t)}$, and the output is a binary label $\in \{0, 1\}$. We train until AUC fails to improve for 10 epochs on 10% of the training set (randomly sampled). We report the best calibration set performance of the following hyperparameters. We use the sklearn defaults otherwise (Pedregosa et al., 2011).

- Layers: 1, 2
- FFN dimension: 32, 64, 128

Transformer The input is a sequence of representations, $h^{(1)} \dots h^{(t)}$ corresponding to thoughts $y_t = y^{(1)} \dots y^{(t)}$. The output is either a binary label $\in \{0, 1\}$ for $\mathbb{P}(\text{correct})$ and $\mathbb{P}(\text{consistent})$, or a sequence of labels $\in \{0, 1\}^t$ for $\mathbb{P}(\text{novel})$ and $\mathbb{P}(\text{leaf})$. For the former, we treat the embeddings as a set (i.e. if *any* representation is sufficient to answer correctly, or be consistent). For the latter, we apply a left-to-right causal attention mask during training, and we use sinusoidal positional encodings to encode the index of each reasoning step. We report the best calibration set performance of the following hyperparameters. In contrast to the linear and MLP models, we find that the Transformer performs best if we *do not* apply PCA and instead operate over the original model dimension.

- Layers: 1, 2
- Model dimension: 16, 32, 64
- FFN dimension: 64, 128
- Number of heads: 4, 8
- Epochs: 5, 10

B.2 LLM experiments

We ran DeepSeek-R1 distilled Qwen 2.5 32B and Llama 70B, and QwQ 32B using lmdeploy (Contributors, 2023) with recommended defaults for each model. lmdeploy natively supports the saving of last layer representations, so it was used for almost all experiments. We ran Qwen 3 32B using vLLM (Kwon et al., 2023) due to early support. Due to computational constraints, we report the mean over a single run.

We downloaded all model weights from `transformers` between April 1, 2025 and May 1, 2025.

Table 1: Probe architecture performance on s1K-1.1 train and calibration splits. Metric: Binary AUROC.

		Linear		MLP		Transformer	
Model	Quantity	Train	Cal	Train	Cal	Train	Cal
DeepSeek-R1 distilled Qwen 2.5 32B	$\mathbb{P}$(correct)	0.936	0.788	0.990	0.779	0.994	0.760
	$\mathbb{P}$(consistent)	0.919	0.788	0.994	0.747	0.991	0.773
	$\mathbb{P}$(leaf)	0.868	0.839	0.936	0.815	0.933	0.852
	$\mathbb{P}$(novel)	0.874	0.686	0.980	0.692	0.896	0.774
DeepSeek-R1 distilled Llama 3.3 70B	$\mathbb{P}$(correct)	0.937	0.765	0.987	0.746	0.991	0.803
	$\mathbb{P}$(consistent)	0.921	0.745	0.994	0.743	0.993	0.748
	$\mathbb{P}$(leaf)	0.864	0.819	0.970	0.802	0.923	0.848
	$\mathbb{P}$(novel)	0.872	0.686	0.981	0.702	0.915	0.774
QwQ 32B	$\mathbb{P}$(correct)	0.943	0.848	0.986	0.838	0.948	0.848
	$\mathbb{P}$(consistent)	0.950	0.699	0.988	0.704	0.939	0.756
	$\mathbb{P}$(leaf)	0.869	0.840	0.942	0.822	0.913	0.857
	$\mathbb{P}$(novel)	0.876	0.677	0.952	0.690	0.895	0.792

C Additional analysis

Figure 7 illustrates the early exit probabilities for each of the three probes. The supervised ("correct") probe reaches high exit probabilities the fastest, but it is also the most overconfident (Figure 3D).

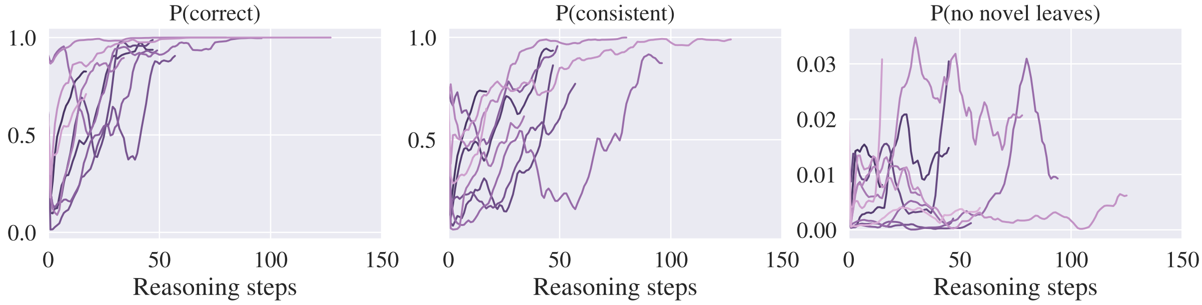


Figure 7: Likelihoods of thought calibration probes over s1K-1.1 test set (10 examples). The "No Leaf" variant is the least monotonic. This could potentially indicate that after reaching the answer, the language model explores new knowledge that is irrelevant to the task.