Reasoning Models Know What's Important, and Encode It in Their Activations

Anonymous Author(s)
Affiliation
Address
email

Abstract

Language models often solve complex tasks by generating long reasoning chains, consisting of many steps with varying importance. While some steps are crucial for generating the final answer, others are removable. Determining which steps matter most, and why, remains an open question central to understanding how models process reasoning. We investigate if this question is best approached through model internals or through tokens of the reasoning chain itself. We find that model activations contain more information than tokens for identifying important reasoning steps. Crucially, by training probes on model activations to predict importance, we show that models encode an internal representation of step importance, even prior to the generation of subsequent steps. This internal representation of importance generalizes across models, is distributed across layers, and does not correlate with surface-level features, such as a step's relative position or its length. Our findings suggest that analyzing activations can reveal aspects of reasoning that surface-level approaches fundamentally miss, indicating that reasoning analyses should look into model internals.

1 Introduction

Reasoning language models solve complex problems by generating long reasoning chains that consist of many intermediate steps (Jaech et al., 2024; Guo et al., 2025; Agarwal et al., 2025; Team et al., 2025). Some of these steps are critical for the final answer, while others are redundant or uninformative, making them removable (Qu et al., 2025). Identifying which steps are truly *important* has significant implications for reasoning efficiency, faithfulness, and interpretability: it determines whether the model's answer is grounded in its reasoning chain, and which steps actually drive the answer.

Identifying important steps can be done by inspecting the tokens of the reasoning chain or by analyzing internal activations. Previous studies have indirectly examined step importance in both ways (Bogdan et al., 2025; Chen et al., 2026a; Levy et al., 2025; Do et al., 2025; Zhang et al., 2025). These approaches are complementary: while token-based approaches are often simpler, they are not always faithful (Lanham et al., 2023) and may lose valuable information in the transition between latent activations and sampled tokens. In this work, we study reasoning step importance using both approaches and compare their utility.

To see if model activations are more indicative of importance compared to tokens, we analyze mathematical reasoning across five models. Following Tutek et al. (2025), we define importance via *removability*: a step is important if removing it prevents the model from reaching the correct answer. For each reasoning chain, we greedily remove steps ranked by either latent influence signals (activation-based) or an external LLM judge (token-based). We find that the activation-based approach consistently identifies a smaller set of steps that is sufficient for generating the

Submitted to 40th Conference on Neural Information Processing Systems (NeurIPS 2026). Do not distribute.

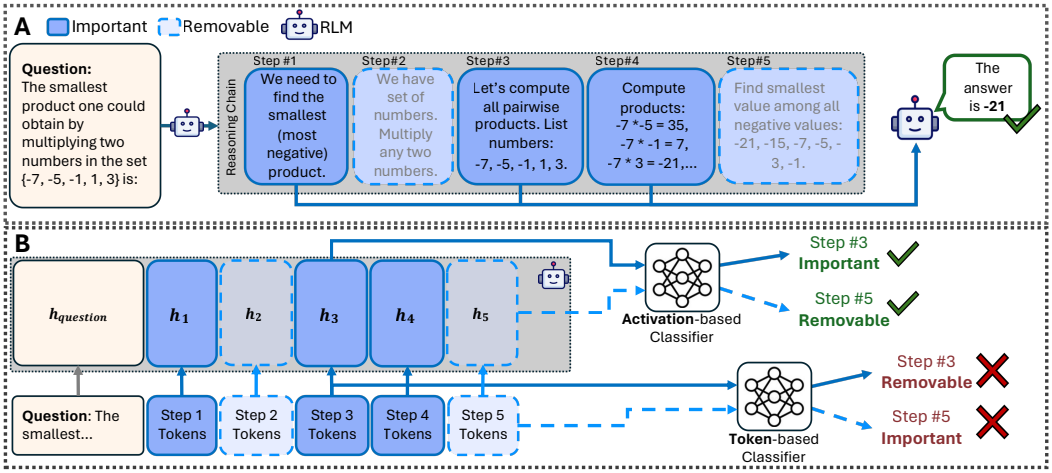


Figure 1: **Overview of reasoning step importance. A.** A reasoning chain can be partitioned into important and removable steps. Important steps make up a sparse subsequence that is sufficient to reach the correct answer. Removable steps can be omitted. **B.** The importance of a reasoning step is encoded in the model's latent activations, and can be extracted using a probe, even before subsequent steps are generated. In contrast, this importance signal can not be accurately distinguished from the reasoning chain alone.

correct answer (Figure 1A), suggesting that latent information leads to more precise identification of reasoning importance.

More interestingly, we ask whether models encode an *internal* notion of step importance. By training probes on latent activations of individual steps, we find that important steps are internally distinguishable from removable ones (Figure 1B), suggesting models represent importance in their activations. This representation relies on activations that integrate the full reasoning history rather than local step content alone. Remarkably, this representation exists even *prior* to the generation of subsequent steps, indicating that reasoning models encode importance pre-hoc. This internal representation generalizes across models: transferring a reasoning trace to a different model yields activations which also encode a similar notion of importance. This internal representation exhibits low correlation with surface-level features, such as position, length and step category, suggesting importance is not surface-level.

Together, our results show that LLMs encode a notion of reasoning step importance within a step's activations—even prior to the generation of subsequent steps—a notion that does not surface in token outputs. This suggests that to better understand different aspects of reasoning in LLMs, one must look beyond tokens and into model internals.

2 Preliminaries

We study the concept of reasoning step importance, and how it is encoded in reasoning models, through the proxy of removability. We consider reasoning steps important if they can not be removed while maintaining the correct answer (see discussion in Appendix A).

2.1 Problem setup

For a prompt x and a model ϕ, we denote $o = \phi(x) = s_1 \circ s_2 \circ \cdots \circ s_n$ as its corresponding reasoning chain of n steps, where $\circ$ denotes concatenation. A reasoning chain subsequence $\mathcal{T} = \{s_{i_k}\} = (s_{i_1}, s_{i_2}, \ldots)$, where $i_k < i_{k+1}$ for all k, is any subsequence of steps of o. We evaluate the quality of reasoning subsequences using *sufficiency* and *attainability*.

Sufficiency measures how well a reasoning subsequence supports generation of a correct final answer. We extend a subsequence $\mathcal{T}$ with an answer-elicitation string σ (e.g., "`So the answer is \boxed{`"; see Appendix B.1) and sample $K = 5$ non-deterministic completions. Sufficiency is the

fraction of completions that yield the ground truth answer a^*:

$$\text{Sufficiency}(\mathcal{T}) = \frac{1}{K}\sum_{k=1}^{K} \mathbf{1}[\phi(x \circ \mathcal{T} \circ \sigma) = a^*] \ . \tag{1}$$

Attainability measures how naturally a reasoning subsequence arises from the model's prior—i.e., how probable it is given the input. We measure this via the perplexity of $\mathcal{T}$ conditioned on x, normalized to $[0, 1]$ and reversed, so that higher scores indicate more probable subsequences. We normalize using two reference bounds. The lower bound is the perplexity of the full chain o, a highly probable continuation of the prompt x. The upper bound is the perplexity of the direct-answer string $\sigma \circ a^*$: the model is "surprised" to see the correct answer immediately after a hard problem statement. Formally:

$$\text{Attainability}(\mathcal{T}) = 1 - \frac{\text{PPL}(\mathcal{T} \mid x) - \text{PPL}(o \mid x)}{\text{PPL}(\sigma \circ a^* \mid x) - \text{PPL}(o \mid x)}, \tag{2}$$

where $\text{PPL}(\cdot \mid x)$ is the perplexity assigned by the model to a sequence given context x.

The two metrics are complementary by design. Sufficiency alone is susceptible to degenerate subsequences: a subsequence consisting solely of the sentence "`The answer is` a^*" would be sufficient, since the model can simply copy the answer—yet such a subsequence is unattainable for a hard problem.

For each reasoning chain, we search for the important steps that make up a **core reasoning subsequence**: a subsequence $\mathcal{T}^*$ that is sufficient, attainable, and as short as possible. Formally, $\mathcal{T}^*$ is the shortest valid subsequence from which no single step can be removed while preserving sufficiency and attainability:

$$\mathcal{T}^* = \arg\min_{\mathcal{T} \subseteq o} |\mathcal{T}| \quad \text{subject to} \quad \text{Sufficiency}(\mathcal{T}) \geq \tau \ \text{ and } \ \text{Attainability}(\mathcal{T}) \geq \tau \ , \tag{3}$$

where $\tau = 0.8$, corresponding to $\frac{4}{5}$ correct answers and low relative perplexity (see App C for other thresholds).

2.2 Experimental details and pre-processing

Models and datasets. We perform our analysis on five reasoning models from several families, differing in pretraining data, architecture, and size: GPT-OSS-20B (Agarwal et al., 2025), DeepSeek-R1-Distill-Qwen (1.5B, 7B, and 14B) (Guo et al., 2025), and OLMo-3-7B-Think (Olmo et al., 2025). We examine the Human Annotated Reasoning Problems (HARP; Yue et al., 2024) and MATH-500 (Hendrycks et al., 2021) datasets, two challenging mathematical reasoning benchmarks containing $4,780$ and 500 problems, respectively.

Generation. For each prompt x, we generate a reasoning chain o using greedy decoding. Following prior work (Lanham et al., 2023; Jacovi et al., 2024; Bogdan et al., 2025), we segment o into steps analogous to sentences. We validate each chain by comparing the final answer $\phi(x)$ to the ground truth a^* using symbol-matching heuristics, since correct mathematical answers may be expressed in multiple equivalent forms. We provide full details in Appendix B.1, with example reasoning chains shown in Appendix F.

Filtering. We restrict our analysis to prompts for which the reasoning chain is both correct and necessary. To measure necessity, we check whether the model can produce a correct answer without reasoning by prompting it with $x \circ \sigma$. We draw $K = 5$ non-deterministic completions and compute the percentage of correct answers. A reasoning chain is considered necessary for a prompt x if the model answers incorrectly at least 4 out of 5 resamples. This ensures that every retained prompt genuinely requires multi-step reasoning. We report the number of filtered prompts in Appendix D.

To reduce the search space of potential important steps in each reasoning chain and account for redundant reasoning paths, we employ two additional pre-processing steps per prompt: we remove shortcut steps and identify a reasoning prefix sufficient to reach the final answer.

Eliminating shortcut steps. A step s_i is considered a *shortcut step* if it alone is sufficient for producing the correct answer, either because $\mathrm{Sufficiency}(s_i)$ is high or because it states the answer directly (e.g., "`So the answer is` -21", see App B.1). Including such steps conflates answer-stating with meaningful reasoning, so we remove them from each chain. We then verify that the remaining chain is still sufficient, and exclude chains that fail this check.

Identifying a sufficient prefix. Reasoning models often reach an answer long before outputting it in the reasoning chain (Mao et al., 2025; Wang et al., 2025). Thus, rather than analyzing the full chain, we restrict our analysis to a *sufficient prefix*: the shortest contiguous subsequence $\mathcal{T}_p = (s_1, s_2, \ldots, s_p)$ that achieves both high sufficiency and attainability. This serves two purposes. First, it discards trivially removable post-answer content (e.g., verification). Second, it ensures that we analyze a single coherent path from the input x to the answer, rather than a chain that may contain multiple redundant sufficient subsequences.

3 Finding important reasoning steps

Having identified a reasoning chain prefix $\mathcal{T}_p$, we now aim to find a core reasoning subsequence $\mathcal{T}^* \subseteq \mathcal{T}_p$. Since exhaustive search over all 2^p subsequences is infeasible, we use a greedy procedure instead. We consider three variants for this procedure: an activation-based variant, a token-based variant, and a randomized variant.

3.1 Methods

The greedy procedure operates in two steps. First, we select steps from $\mathcal{T}_p$ that have direct or indirect influence on the answer, forming $\mathcal{T}_{\mathrm{inf}}$. Second, we rank steps in $\mathcal{T}_{\mathrm{inf}}$ and iteratively remove them, while maintaining sufficiency and attainability of the resulting subsequence, yielding the core subsequence $\mathcal{T}^*$. The three variants differ in how they measure influence in the first stage and rank steps in the second stage.

Activation-based variant. To identify the steps that influence the answer, we calculate the influence between every pair of steps s_i, s_j, $j > i$. We do so with gradient attribution (Li et al., 2016, ;see Appendix E.1). We sum the probabilities of all tokens in s_j and compute the gradient of this sum with respect to the input embeddings of all tokens in s_i. The influence score is the Frobenius norm of the resulting gradient tensor:

$$\mathbf{I}(s_i, s_j) = \left\| \frac{\partial \sum_{t \in s_j} \mathbb{P}_t}{\partial \mathbf{E}_{s_i}} \right\|_F, \tag{4}$$

where $\mathbf{E}_{s_i}$ is the matrix of input embeddings for step s_i and $\mathbb{P}_t$ is the probability of token t. We set the influence of a step on itself to zero. This yields an influence matrix $\mathbf{I}$ over all pairs of steps (s_i, s_j) with $j > i$, which we compute in a single backward pass per target step.

In the first stage, we use $\mathbf{I}$ to identify steps that have direct or indirect influence on the answer. We initialize $\mathcal{T}_{\mathrm{inf}} = (s_p)$ (s_p is the required final prefix step; see Section 2). For each $s_j \in \mathcal{T}_{\mathrm{inf}}$, we recursively add all predecessor steps s_i whose cumulative influence $\sum_{s_i} \mathbf{I}(s_i, s_j)$ exceeds a threshold. We sweep over thresholds $\{0.1, 0.2, \ldots, 1.0\}$, selecting the lowest one such that $\mathcal{T}_{\mathrm{inf}}$ maintains high sufficiency and attainability.

In the second stage, we rank the remaining steps in $\mathcal{T}_{\mathrm{inf}}$ by their mean influence on subsequent steps. We attempt removals from lowest to highest rank, and retain a removal only when the resulting subsequence still satisfies the sufficiency and attainability thresholds. The full implementation of the activation-based approach is presented in Appendix B.2.1.

Token-based variant. Rather than using internal model signals, the token-based method relies entirely on an external LLM (Gemini-2.5-Pro (Comanici et al., 2025)). In the first stage, we prompt the LLM to identify which steps in $\mathcal{T}_p$ influence the answer, yielding $\mathcal{T}_{\mathrm{inf}}$. In the second stage, we prompt the LLM to rank the steps in $\mathcal{T}_{\mathrm{inf}}$ by importance and use this ranking in place of the mean influence score. The prompts used in this variant are presented in Appendix B.2.2.

Table 1: **Average core reasoning subsequence size (%, lower is better).** We measure the average size of the core reasoning subsequence $\mathcal{T}^*$, as a percentage of the sufficient reasoning prefix $\mathcal{T}_p$. Across models and datasets, the activation-based method produces smaller subsequences (underlined), suggesting it better identifies the steps that matter most.

Dataset	Method	GPT-OSS	DS-1.5	DS-7	DS-14	Olmo3
HARP	Random Greedy	26.2	29.7	29.2	32.1	<u>24.1</u>
	Token-based	31.7	41.0	41.4	40.7	35.3
	Activation-based	<u>23.0</u>	<u>26.4</u>	<u>27.0</u>	<u>28.1</u>	26.3
Math-500	Random Greedy	33.4	26.0	24.8	30.0	21.8
	Token-based	42.5	31.4	35.0	41.6	34.1
	Activation-based	<u>30.8</u>	<u>22.1</u>	<u>23.1</u>	<u>27.8</u>	<u>18.0</u>

Random variant. We randomly rank steps for removal. This variant serves as a strong baseline: any improvement of the other variants over it can be attributed to the quality of their respective rankings. We skip the first stage, noting $\mathcal{T}_{\text{inf}} = \mathcal{T}_p$. In the second stage, steps in $\mathcal{T}_{\text{inf}}$ are ranked in a random order, rather than by a meaningful signal.[1]

3.2 Results

We evaluate each variant by the size of the core reasoning subsequence $\mathcal{T}^*$ it produces. Results in Table 1 show that the activation-based variant consistently yields the smallest core subsequences. Detailed per-prompt comparisons in Appendix E.2 also point to a consistent advantage of the activation-based variant.We therefore adopt the activation-based approach for all subsequent analyses, using it to assign binary labels to steps in $\mathcal{T}_p$: steps removed in the first stage are labeled *removable*, and steps retained after the second stage are labeled important and *non-removable*. Steps removed during the second stage are left unlabeled; see Appendix A for a discussion on this choice.

4 Detecting the notion of importance

Having found core reasoning subsequences, we now assess if we can distinguish important from removable reasoning steps using their representations. Specifically, we ask if models maintain an internal notion of importance, and if so, what kind of representation, from tokens to context-aware activations, is sufficient to detect it?

4.1 Models encode importance in their activations

We begin by analyzing if models maintain an internal notion of importance in their activations. To test this, we train a binary classification probe on the activations of each reasoning step to predict if it is removable, using labels derived from the best-performing activation-based variant in Section 3. We pass full prompts, including the input problem and the sufficient prefix, through the model. We extract activations at the positions of each target reasoning step. The activations of a step form a tensor of shape $S \times L \times d$, where S, L, and d are the number of tokens, layers, and the hidden dimension, respectively. We average the activation tensor over the token dimension and flatten it over the layers and hidden dimensions. As the probe, we use a two-layer MLP with a hidden dimension of 128, and a ReLU non-linearity; alternative architectures are evaluated in App B.3.1. Probes are trained and evaluated on a balanced set of removable and non-removable steps on 3 random splits.

We find that activation-based probes (red bars in Figure 2) yield high ($\geq 80\%$) accuracies, indicating that models internally encode a signal for reasoning step importance. Notably, since a step's activations contain information only from that step and preceding steps (due to causal attention), this notion of importance is *pre-hoc*: it exists before the model generates the subsequent steps that rely on the current step and thereby determine its importance.

[1] Because all steps are removal candidates, this variant requires significantly more runtime.

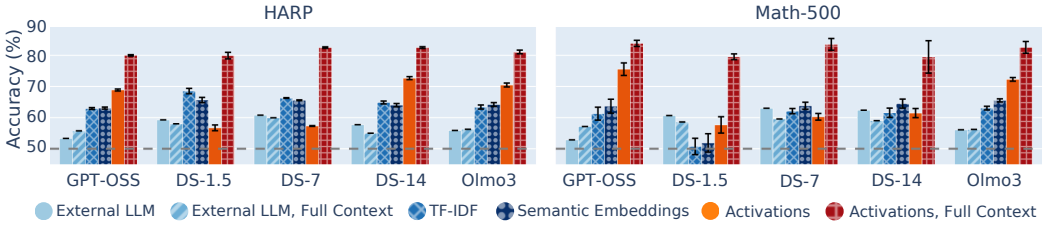


Figure 2: **Importance detection requires full context and strong representations.** We compare methods for predicting reasoning step importance pre-hoc, ranging from token-based approaches to full-context activations. Token-based LLM-as-a-judge, with or without full context, reaches low accuracies ($\leq 60\%$). Methods using context-free representations (TF-IDF, semantic embeddings, and context-free activations) perform moderately. Full-context activation probes achieve the highest accuracies ($\geq 80\%$), confirming that both full context and rich representations are necessary to accurately detect reasoning step importance.

4.2 Detecting importance requires a strong representation and context

Having established that full-context activation probes successfully identify importance pre-hoc, we investigate whether simpler extraction methods can suffice. The baselines we consider differ along two dimensions: context availability and the representational capacity—the richness of each reasoning step's encoding.

On the context side, we evaluate a context-free variant of the activation probe, in which only the target step is encoded by the model without any preceding reasoning. To measure if importance can be detected from weaker representations of reasoning steps, we extend the context-free evaluation to consider methods with varying encoder capacity. We evaluate two standard text classification approaches. In both approaches we represent each reasoning step as a vector and train a classifier to predict step importance. The classifier's architecture is identical to the activation probe, so any difference in performance reflects representation quality rather than classifier capacity. First, we use semantic embeddings: we pass each step through a pre-trained sentence transformer (`all-MiniLM-L6-v2`; Reimers & Gurevych, 2019), resulting in a vector that captures sentence-level semantics. Second, we represent each step as a TF-IDF vector (Sparck Jones, 1972) over n-grams ($n \in [1, 5]$), capturing surface lexical patterns. Finally, we evaluate a strong external LLM (Gemini-2.5-Pro) as a zero-shot judge in both context-free and full-context settings. We prompt it to classify each reasoning step[2] as removable or non-removable, using classification prompts shown in Appendix B.3.2. This method operates purely over tokens as a representation of reasoning steps.

The results (Figure 2) are consistent across models and datasets. Removing context from the activation probe leads to a substantial drop in accuracy (10–20%, orange bar), establishing that the input prompt and the preceding reasoning chain are necessary to identify step importance. This suggests that methods that encode reasoning steps in isolation (Bogdan et al., 2025; Levy et al., 2025) might miss such signals. Additionally, decreasing the representational capacity often harms the ability to detect reasoning step importance, as evidenced by the semantic embeddings and TF-IDF approaches. Finally, the LLM-as-a-judge reaches low accuracies ($\leq 60\%$), even when provided with full context. This indicates that token-level access alone is likely insufficient, as even a strong reasoning model does not perform the task well without access to latent representations.

Taken together, these results confirm that accurately detecting reasoning step importance requires access to both strong latent representations and full context. Both requirements are satisfied in the activations of the model that generated the reasoning chain, further motivating the use of full-context activations for investigating reasoning.

[2]Due to budget constraints, LLM-as-a-judge evaluations use 1500 steps per model and dataset. Results of other methods on this subset are consistent with full comparisons (Appendix E.4).

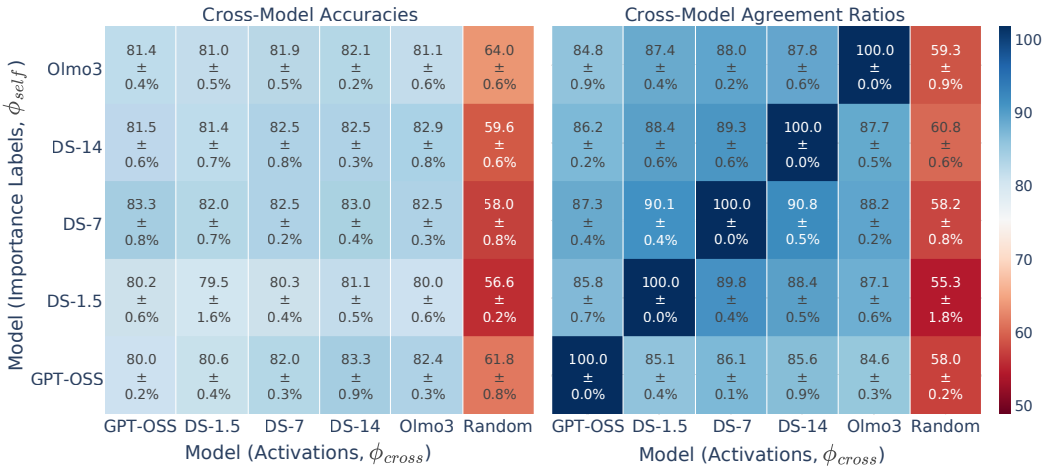


Figure 3: **Activation-based probes (on HARP) perform similarly across models. Left:** Probes trained on activations from one model (ϕ_{cross}, x axis) to predict importance labels from another (ϕ_{self}, y axis) achieve high accuracy across all pairs. **Right:** We train two activation probes, on activations from ϕ_{cross} and ϕ_{self}, both with labels from ϕ_{self}. The fraction of reasoning steps assigned the same label by both probes is high across all pairs, suggesting importance is encoded consistently across models. The *random* columns report results of a randomly initialized model matching ϕ_{self} to verify that probe performance depends mostly on informative activation features rather than spurious correlations.

5 Analyzing the notion of importance

Having established that importance is encoded in full-context model activations, we now analyze the nature of this encoding: how does it vary across models, where is it encoded in activations, and which other features does it correlate with.

5.1 Importance Is cross-model

Prior work has shown that reasoning chains can generalize across models (Pal et al., 2026), yet others posited that models explain their own computations better than other models can (Li et al., 2025). We ask if the internal notion of importance generalizes across models: can a probe trained on activations from one model predict importance labels from another?

For every pair of models ϕ_{self}, ϕ_{cross}, we pass reasoning chains generated by ϕ_{self} through ϕ_{cross} to obtain activation tensors. We train a probe on activations from ϕ_{cross} to predict importance labels of ϕ_{self}, using a two-layer MLP probe trained on full context activations. We train and evaluate over 3 random splits across all labeled reasoning steps from ϕ_{self}. As a baseline, we replace ϕ_{cross} with a randomly initialized model matching the architecture of ϕ_{self} (Tenney et al., 2019), and train a probe on its activations following the same procedure.

For each model pair, we measure cross-model probe accuracy and agreement ratio. The agreement ratio is the percentage of steps classified identically by the probe trained on ϕ_{self} activations and the probe trained on ϕ_{cross} activations. We report results for HARP (Figure 3) and MATH-500 (Appendix E.5). Across all model pairs, the cross-model probe accuracy matches the within-model setting, and agreement ratios are high. In contrast, the randomized baseline reaches lower but non-random accuracy, confirming that probe performance *mainly* relies on meaningful activation structure, which does not appear in randomized weights. Taken together, these results indicate that the importance of a reasoning step is consistently encoded in activations regardless of which reasoning model processes it, suggesting importance is not model-specific but a more universal concept.

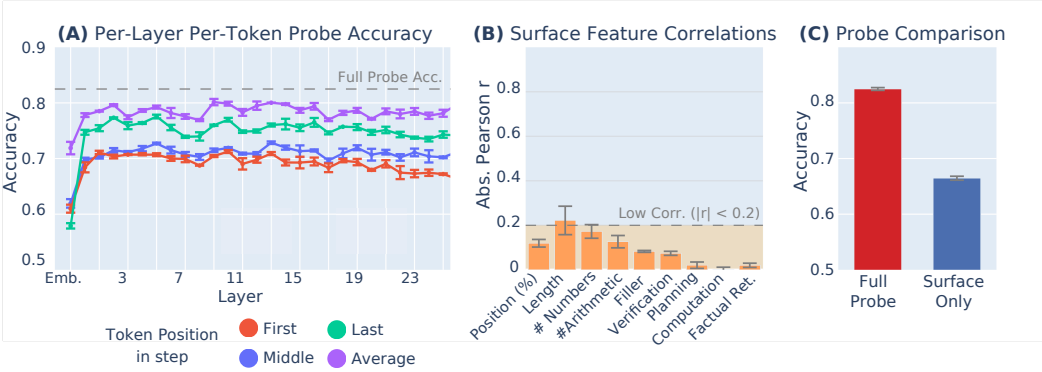


Figure 4: **Analysis of the importance probe in DeepSeek-7B (HARP dataset). (A)** We train separate probes on residual activations at each layer and token position; no single layer or position matches the accuracy of the full probe trained on all layers. **(B)** Individual surface-level features show low correlation with probe predictions; **(C)** a regression model trained on all surface features (combined) performs substantially worse than the full probe, indicating that the probe captures meaningful structure beyond surface-level information.

5.2 Importance is represented by distributed, non-surface-level features

We analyze the importance probe by examining two questions: is importance localized to specific layers and positions, and whether it reduces to surface-level properties of the text.

Importance is distributed across layers and positions. To isolate where importance is encoded, we train a separate probe on the residual-stream activations of each individual layer, at specific token positions of each step (first, middle, and last). We also train a per-layer probe over activations averaged across positions. Figure 4a shows representative results for DeepSeek-7B on the HARP dataset (similar results for other models in Appendix E.6). Probes trained on individual layers achieve similar accuracy to one another, indicating that importance is not localized to specific layers. Moreover, no single-layer or single-position probe matches the accuracy of the full probe, which pools information across all layers and positions. This indicates that the features underlying importance are distributed throughout the activations: individual layers capture substantial signal, but combining them is necessary to correctly classify the most ambiguous reasoning steps.

Importance is not a function of surface-level features. We verify that the probe does not only recover surface-level features, which we define as those derivable from tokens. We evaluate a set of candidates: relative position of the step within the reasoning chain, step length in tokens, number of numerical and arithmetic tokens, and features that check whether specific categories of expressions appear in the step (e.g. "Let's think." indicates the "filler" category; see Appendix B.4 for the full list). Individually, all features show low correlation with the probe's predictions (Figure 4b). Combining these features, a linear regression model trained on the full feature set performs better than random, but substantially worse than the activation-based probe (Figure 4c). Similar results for other models are shown in Appendix E.6. This suggests that while surface-level features can help in identifying importance, our activation-based probe identifies a richer notion of importance that accurately predicts more steps.

Together, these results suggest that the probe captures a notion of importance that is distributed across the model and irreducible to surface-level token statistics. This further indicates that full-context, full-layers latent activations are a better representation to study reasoning importance, and potentially other latent notions.

6 Related work

Interpreting reasoning chains. Prior work has studied reasoning chains from multiple perspectives, mostly relying on tokens: categorizing steps by relevance, coherence, and correctness (Jacovi et al., 2024; Do et al., 2025); analyzing reasoning structure as a graph of semantically-clustered steps

(Teng et al., 2025; Feng et al., 2025; Jiang et al., 2025); or identifying influential steps by sampling replacements (Bogdan et al., 2025). However, analysis relying on tokens has well-known limitations: generated reasoning does not always reflect model internals (Agarwal et al., 2024; Stechly et al., 2025), and many steps are "performative" and have minimal effect on the final answer (Zhang et al., 2025; Zhao et al., 2025; Boppana et al., 2026). This limitation led to studies examining reasoning through model latents directly, via the logit lens (nostalgebraist, 2020; Chen et al., 2026b), algorithmic primitives (Lippl et al., 2025), or geometric patterns (Zhou et al., 2025; Minegishi et al., 2025). Where these studies either predict correctness based on perceived thinking depth or offer high level perspectives of latent reasoning, we study the concept of importance of reasoning steps and whether it can be decoded from tokens or model latents.

Probing for abstract concepts. Probing classifiers (Alain & Bengio, 2016; Belinkov, 2022) have been used to identify abstract concepts encoded in language model representations, including downstream performance (Afzal et al., 2025; Zhu et al., 2022), truthfulness (Joshi et al., 2023; Simhi et al., 2024; Orgad et al., 2024), and model beliefs (Bortoletto et al., 2024) and attributes (Arditi et al., 2024; Papadatos & Freedman, 2024). While these probes are often linear and targeted at specific positions, cross-layer probes have been shown to yield strong results (Bar-Shalom et al., 2025). Building on these, we train probes to identify step importance in reasoning chains, revealing that this abstract notion is encoded in model activations even prior to generation of subsequent steps.

Importance estimation for reasoning chain pruning. Measuring reasoning step importance can be used for pruning reasoning chains and obtaining a minimal sufficient reasoning trace (Ding et al., 2024). Along this line, several studies prune reasoning chain in place, removing tokens or entire steps by using signals from step entropy or perplexity (Cui et al., 2025; Li et al., 2026; Xiao et al., 2025), token attribution (Wang et al., 2026), prompting (Liu et al., 2024), or external verifiers (Lin et al., 2025). While these studies implicitly raise the question of *"which reasoning steps are important?"*, we do so explicitly: our focus is not pruning itself, but rather using step removability as a causal intervention to investigate if and how LLMs encode the concept of step importance.

7 Conclusions, discussion and limitations

Do reasoning models "know" when a reasoning step is important? We approached this question through the lens of removability—whether a step can be removed while still leaving an attainable path to the correct answer. Our findings show that most reasoning steps are not important, and that model internals contain signals that accurately identify the most important steps. These signals are best recoverable from full-context latent activations. They are distributed across layers and positions, and transferable across models.

Our results have several implications. First, model activations encode abstract concepts that might not surface at the token level. This suggests a form of unfaithfulness, where steps that appear significant in the generated trace are in fact removable. We thus argue that studying reasoning through full-context model activations offers a more faithful lens on the reasoning process. Second, if a small subsequence of steps is both sufficient for reaching the correct answer and highly attainable, then this core reasoning subsequence—rather than the full trace—may be a more informative target for monitoring and studying internal computation.

Several of our analyses yielded negative yet worthwhile results. Our attempt at analyzing the importance probe weights with standard tools (e.g., logit lens; nostalgebraist, 2020) produced null findings; attempts to taxonomize core reasoning steps by their functionality were similarly inconclusive. We further discuss these negative results in Appendix E.7.

Our work has several limitations. Our analysis focuses on mathematical reasoning, which is a standard benchmark for evaluating reasoning models. Extending this analysis to other domains where reasoning models excel, such as coding or open-ended problems, and validating our findings hold there, is a natural direction for further work. Finally, while we demonstrate that models encode a concept of importance in their activations, we do not use this signal to fine-tune models on core reasoning chains; this is beyond the analytic focus of this work. Nonetheless, grounded in our notion of attainability and in line with the pruning literature (Section 6), we believe such fine-tuning is a viable and promising direction.

References

Anum Afzal, Florian Matthes, Gal Chechik, and Yftah Ziser. Knowing before saying: LLM representations encode information about chain-of-thought success before completion. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 12791–12806, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.662. URL https://aclanthology.org/2025.findings-acl.662/.

Chirag Agarwal, Sree Harsha Tanneru, and Himabindu Lakkaraju. Faithfulness vs. plausibility: On the (un)reliability of explanations from large language models. *CoRR*, abs/2402.04614, 2024. doi: 10.48550/ARXIV.2402.04614. URL https://doi.org/10.48550/arXiv.2402.04614.

Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K Arora, Yu Bai, Bowen Baker, Haiming Bao, et al. gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925*, 2025.

Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes. *arXiv preprint arXiv:1610.01644*, 2016.

Andy Arditi, Oscar Obeso, Aaquib Syed, Daniel Paleka, Nina Panickssery, Wes Gurnee, and Neel Nanda. Refusal in language models is mediated by a single direction. *Advances in Neural Information Processing Systems*, 37:136037–136083, 2024.

Guy Bar-Shalom, Fabrizio Frasca, Yaniv Galron, Yftah Ziser, and Haggai Maron. Beyond token probes: Hallucination detection via activation tensors with act-vit. *arXiv preprint arXiv:2510.00296*, 2025.

Yonatan Belinkov. Probing classifiers: Promises, shortcomings, and advances. *Computational Linguistics*, 48(1):207–219, 2022.

Paul C. Bogdan, Uzay Macar, Neel Nanda, and Arthur Conmy. Thought anchors: Which llm reasoning steps matter?, 2025. URL https://arxiv.org/abs/2506.19143.

Siddharth Boppana, Annabel Ma, Max Loeffler, Raphael Sarfati, Eric Bigelow, Atticus Geiger, Owen Lewis, and Jack Merullo. Reasoning theater: Disentangling model beliefs from chain-of-thought. *arXiv preprint arXiv:2603.05488*, 2026.

Matteo Bortoletto, Constantin Ruhdorfer, Lei Shi, and Andreas Bulling. Brittle minds, fixable activations: Understanding belief representations in language models. *arXiv preprint arXiv:2406.17513*, 2024.

Beiduo Chen, Tiancheng Hu, Caiqi Zhang, Robert Litschko, Anna Korhonen, and Barbara Plank. Decoupling the effect of chain-of-thought reasoning: A human label variation perspective, 2026a. URL https://arxiv.org/abs/2601.03154.

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*, 2026b.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

Yingqian Cui, Pengfei He, Jingying Zeng, Hui Liu, Xianfeng Tang, Zhenwei Dai, Yan Han, Chen Luo, Jing Huang, Zhen Li, Suhang Wang, Yue Xing, Jiliang Tang, and Qi He. Stepwise perplexity-guided refinement for efficient chain-of-thought reasoning in large language models. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 18581–18597, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.956. URL https://aclanthology.org/2025.findings-acl.956/.

Mengru Ding, Hanmeng Liu, Zhizhang Fu, Jian Song, Wenbo Xie, and Yue Zhang. Break the chain: Large language models can be shortcut reasoners. *arXiv preprint arXiv:2406.06580*, 2024.

Heejin Do, Jaehui Hwang, Dongyoon Han, Seong Joon Oh, and Sangdoo Yun. What defines good reasoning in llms? dissecting reasoning steps with multi-aspect evaluation, 2025. URL https://arxiv.org/abs/2510.20603.

Yunzhen Feng, Julia Kempe, Cheng Zhang, Parag Jain, and Anthony Hartshorn. What characterizes effective reasoning? revisiting length, review, and structure of cot. *CoRR*, abs/2509.19284, 2025. doi: 10.48550/ARXIV.2509.19284. URL https://doi.org/10.48550/arXiv.2509.19284.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

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

Alon Jacovi, Yonatan Bitton, Bernd Bohnet, Jonathan Herzig, Or Honovich, Michael Tseng, Michael Collins, Roee Aharoni, and Mor Geva. A chain-of-thought is as strong as its weakest link: A benchmark for verifiers of reasoning chains. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 4615–4634, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.254. URL https://aclanthology.org/2024.acl-long.254/.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

Gangwei Jiang, Yahui Liu, Zhaoyi Li, Wei Bi, Fuzheng Zhang, Linqi Song, Ying Wei, and Defu Lian. What makes a good reasoning chain? uncovering structural patterns in long chain-of-thought reasoning. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 6490–6514, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.329. URL https://aclanthology.org/2025.emnlp-main.329/.

Nitish Joshi, Javier Rando, Abulhair Saparov, Najoung Kim, and He He. Personas as a way to model truthfulness in language models. *ArXiv*, abs/2310.18168, 2023. URL https://api.semanticscholar.org/CorpusID:264555113.

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.

Mosh Levy, Zohar Elyoseph, and Yoav Goldberg. Humans perceive wrong narratives from ai reasoning texts. *arXiv preprint arXiv:2508.16599*, 2025.

Belinda Z Li, Zifan Carl Guo, Vincent Huang, Jacob Steinhardt, and Jacob Andreas. Training language models to explain their own computations. *arXiv preprint arXiv:2511.08579*, 2025.

Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. Visualizing and understanding neural models in nlp. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 681–691, 2016.

Zeju Li, Jianyuan Zhong, Ziyang Zheng, Xiangyu Wen, Zhijian Xu, Yingying Cheng, Fan Zhang, and Qiang Xu. Making slow thinking faster: Compressing llm chain-of-thought via step entropy. In *The Fourteenth International Conference on Learning Representations*, 2026.

Weizhe Lin, Xing Li, Zhiyuan Yang, Xiaojin Fu, Hui-Ling Zhen, Yaoyuan Wang, Xianzhi Yu, Wulong Liu, Xiaosong Li, and Mingxuan Yuan. Trimr: Verifier-based training-free thinking compression for efficient test-time scaling. *arXiv preprint arXiv:2505.17155*, 2025.

Samuel Lippl, Thomas McGee, Kimberly Lopez, Ziwen Pan, Pierce Zhang, Salma Ziadi, Oliver Eberle, and Ida Momennejad. Algorithmic primitives and compositional geometry of reasoning in language models. *CoRR*, abs/2510.15987, 2025. doi: 10.48550/ARXIV.2510.15987. URL https://doi.org/10.48550/arXiv.2510.15987.

Tengxiao Liu, Qipeng Guo, Xiangkun Hu, Cheng Jiayang, Yue Zhang, Xipeng Qiu, and Zheng Zhang. Can language models learn to skip steps? In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 45359–45385. Curran Associates, Inc., 2024. doi: 10.52202/079017-1441. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/504fa7e518da9d1b53a233ed20a38b46-Paper-Conference.pdf.

Minjia Mao, Bowen Yin, Yu Zhu, and Xiao Fang. Early stopping chain-of-thoughts in large language models. *arXiv e-prints*, pp. arXiv–2509, 2025.

Gouki Minegishi, Hiroki Furuta, Takeshi Kojima, Yusuke Iwasawa, and Yutaka Matsuo. Topology of reasoning: Understanding large reasoning models through reasoning graph properties, 2025. URL https://arxiv.org/abs/2506.05744.

nostalgebraist. Interpreting GPT: The logit lens, 2020. URL https://www.lesswrong.com/posts/AcKRB8wDpdaN6v6ru/interpreting-gpt-the-logit-lens.

Team Olmo, Allyson Ettinger, Amanda Bertsch, Bailey Kuehl, David Graham, David Heineman, Dirk Groeneveld, Faeze Brahman, Finbarr Timbers, Hamish Ivison, et al. Olmo 3. *arXiv e-prints*, pp. arXiv–2512, 2025.

Hadas Orgad, Michael Toker, Zorik Gekhman, Roi Reichart, Idan Szpektor, Hadas Kotek, and Yonatan Belinkov. Llms know more than they show: On the intrinsic representation of llm hallucinations. *arXiv preprint arXiv:2410.02707*, 2024.

Koyena Pal, David Bau, and Chandan Singh. Do explanations generalize across large reasoning models?, 2026. URL https://arxiv.org/abs/2601.11517.

Henry Papadatos and Rachel Freedman. Linear probe penalties reduce llm sycophancy. *arXiv preprint arXiv:2412.00967*, 2024.

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. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Xiaoye Qu, Yafu Li, Zhao yu Su, Weigao Sun, Jianhao Yan, Dongrui Liu, Ganqu Cui, Daizong Liu, Shuxian Liang, Junxian He, Peng Li, Wei Wei, Jing Shao, Chaochao Lu, Yue Zhang, Xian-Sheng Hua, Bowen Zhou, and Yu Cheng. A survey of efficient reasoning for large reasoning models: Language, multimodality, and beyond. *ArXiv*, abs/2503.21614, 2025. URL https://api.semanticscholar.org/CorpusID:277349297.

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019. URL http://arxiv.org/abs/1908.10084.

Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *International conference on machine learning*, pp. 3145–3153. PMlR, 2017.

Adi Simhi, Jonathan Herzig, Idan Szpektor, and Yonatan Belinkov. Constructing benchmarks and interventions for combating hallucinations in llms. *arXiv preprint arXiv:2404.09971*, 2024.

Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 28(1):11–21, 1972.

Kaya Stechly, Karthik Valmeekam, Atharva Gundawar, Vardhan Palod, and Subbarao Kambhampati. Beyond semantics: The unreasonable effectiveness of reasonless intermediate tokens. *CoRR*, abs/2505.13775, 2025. doi: 10.48550/ARXIV.2505.13775. URL https://doi.org/10.48550/arXiv.2505.13775.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks, 2017. URL https://arxiv.org/abs/1703.01365.

Kimi Team, Yifan Bai, Yiping Bao, Guanduo Chen, Jiahao Chen, Ningxin Chen, Ruijue Chen, Yanru Chen, Yuankun Chen, Yutian Chen, et al. Kimi k2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*, 2025.

Fengwei Teng, Quan Shi, Zhaoyang Yu, Jiayi Zhang, Yuyu Luo, Chenglin Wu, and Zhijiang Guo. Atom of thoughts for markov llm test-time scaling. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R Thomas McCoy, Najoung Kim, Benjamin Van Durme, Samuel R Bowman, Dipanjan Das, et al. What do you learn from context? probing for sentence structure in contextualized word representations. *arXiv preprint arXiv:1905.06316*, 2019.

Martin Tutek, Fateme Hashemi Chaleshtori, Ana Marasović, and Yonatan Belinkov. Measuring chain of thought faithfulness by unlearning reasoning steps, 2025. URL https://arxiv.org/abs/2502.14829.

Siyuan Wang, Yanchen Liu, and Xiang Ren. Segment-level attribution for selective learning of long reasoning traces. *arXiv preprint arXiv:2602.00425*, 2026.

Zezhong Wang, Xingshan Zeng, Weiwen Liu, Yufei Wang, Liangyou Li, Yasheng Wang, Lifeng Shang, Xin Jiang, Qun Liu, and Kam-Fai Wong. Chain-of-probe: Examining the necessity and accuracy of cot step-by-step. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 2586–2606, 2025.

Yang Xiao, Jiashuo Wang, Ruifeng Yuan, Chunpu Xu, Kaishuai Xu, Wenjie Li, and Pengfei Liu. Limopro: Reasoning refinement for efficient and effective test-time scaling. *arXiv preprint arXiv:2505.19187*, 2025.

Albert S Yue, Lovish Madaan, Ted Moskovitz, DJ Strouse, and Aaditya K Singh. Harp: A challenging human-annotated math reasoning benchmark. *arXiv e-prints*, pp. arXiv–2412, 2024.

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

Jiachen Zhao, Yiyou Sun, Weiyan Shi, and Dawn Song. Can aha moments be fake? identifying true and decorative thinking steps in chain-of-thought. *arXiv preprint arXiv:2510.24941*, 2025.

Yufa Zhou, Yixiao Wang, Xunjian Yin, Shuyan Zhou, and Anru R Zhang. The geometry of reasoning: Flowing logics in representation space. *arXiv preprint arXiv:2510.09782*, 2025.

Zining Zhu, Soroosh Shahtalebi, and Frank Rudzicz. Predicting fine-tuning performance with probing. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 11534–11547, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.793. URL https://aclanthology.org/2022.emnlp-main.793/.

A Discussion on removability

Motivation for using step removability as a proxy for importance. We investigate whether LLMs latently encode a notion of reasoning step importance. To operationalize importance, we adopt a causal perspective: a step is important to the extent that intervening on it affects the model's ability to process the reasoning chain and produce a correct answer. The first practical choice is the granularity of the intervention: intervening on a single token or a single step in isolation is often not enough to induce a measurable effect (see Appendix E.7), so we operate at the level of reasoning subsequences. The second choice is the nature of the intervention itself. One can remove a step, substitute it with a paraphrase, an uninformative filler, or an incorrect alternative (Lanham et al., 2023). However, applying substitution-based interventions across multiple steps is problematic. It risks entangling the intervention with uncontrolled confounders of the replacement process, making it difficult to attribute downstream behavioral changes solely to targeted steps. Further more, this introduces generative overhead, requiring a capable model to perform consistent replacements. We therefore adopt step removal: it is simple to apply in parallel across multiple steps and has a direct effect on downstream reasoning. Furthermore, it can have practical implications for reasoning efficiency: if removability can be accurately detected from latent activations, the probes that detect it could be used at inference time to omit removable steps and reduce context length.

Alternative definition of removable steps. In Section 3, we define removable steps as those removed during the first stage of identifying the core reasoning subsequence $\mathcal{T}^*$. An alternative would be to treat any step not in $\mathcal{T}^*$ as removable, additionally including steps eliminated during the second stage ($\mathcal{T}_{\text{inf}} \to \mathcal{T}^*$). We opt against this broader definition because step importance is not binary but continuous: steps removed during the first stage occupy the low-importance extreme, steps in $\mathcal{T}^*$ occupy the high-importance extreme, and the remaining steps fall somewhere in between. We restrict our analysis to steps at these two extremes. This yields the cleanest "ground-truth" labels for our classification experiments in Section 4.

B Technical details

We provide technical details for each part in the experimentation process.

B.1 Reasoning chain generation, step segmentation and evaluation

Reasoning chain generation. For each input prompt x, we generate a single deterministic reasoning chain of up to $8,000$ tokens using greedy decoding. This limit is imposed for several reasons. First, to maintain efficiency, given the high volume of analyzed prompts (See Appendix D). Second, we found that across models, increasing the maximal chain size only led to a slight increase in model accuracy. Thus we chose this limit as a compromise. For the HARP dataset, which is divided into 6 difficulty levels, we analyze a range of difficulties fit to each model's capability. The two strongest models, GPT-OSS-20B and DeepSeek-R1-Qwen-14B, are evaluated across all difficulty levels. DeepSeek-R1-Qwen-1.5B, DeepSeek-R1-Qwen-7B and Olmo-3-7B-Think are evaluated on difficulties 1–4. For the MATH-500 dataset, all 500 prompts are analyzed across all models. For GPT-OSS-20B, we use the default "medium" reasoning depth setting.

Segmentation into reasoning steps. To segment a reasoning chain into steps, we follow standard procedure (Bogdan et al., 2025; Zhang et al., 2025; Minegishi et al., 2025), splitting at paragraph-ending patterns (two consecutive newline characters) or sentence-ending patterns (a period, question mark, or exclamation mark followed by a space or newline).

Evaluation. When forcing a model to generate an answer after a given reasoning subsequence (as part of sufficiency evaluation), we append a model-specific suffix σ to elicit the final response. For DeepSeek and Olmo models, this takes the form:

$$\sigma = \text{``So the answer is \textbackslash boxed\{''}, \tag{5}$$

whereas for GPT-OSS we use:

$$\sigma = \text{``<|end|><|start|>final<|message|>The answer is \textbackslash boxed\{''}. \tag{6}$$

We chose these specific suffixes to best match the way these models generate a final answer during standard generation.

To evaluate sufficiency and necessity, we sample 5 different completions from the model. We use a sampling temperature of 0.6 and top_p $= 0.95$.

Throughout the analysis, we use fixed thresholds of $\text{Sufficiency} = 0.8$ for sufficiency, $\text{Attainability} = 0.8$ for attainability (see discussion and ablations in Appendix C).

Answer comparisons. In mathematical reasoning, the same answer can be represented in different ways. For instance, the answer $-10 \leq x \leq 1$ is equivalent to $x \in [-10, 1]$; just as $5+x$ and $x+5$ are identical expressions. Thus, when a model completes a given answer, the comparison to the ground truth answer must take these representations into account. To account for these cases, we compare generated answers against ground truth using a suite of heuristics designed to handle the variety of forms in which the results may appear, including tuples, ranges, fractions, numeric quantities, and symbolic expressions. Full implementation details are available in our code. It is also noteworthy that this discrepancy limits us from directly using token probabilities in our analysis.

Eliminating shortcut steps. Reasoning chains often contain steps that, when viewed in isolation, can act as potential shortcuts to the final answer. Such shortcuts can harm the validity of our sufficiency measure for any reasoning chain that contains them. To prevent these shortcuts from confounding our analysis, we remove these steps from the generated chains prior to the identification of a sufficient prefix (Section 2) and core reasoning subsequence (Section 3). To identify shortcut steps, we measure the sufficiency of each individual step separately (i.e. we measure $\text{Sufficiency}(\mathcal{T})$ when $\mathcal{T} = (s_i)$ for each step s_i). We consider a step to be a shortcut step if it answers at least one of the following criteria:

- it alone is sufficient to reach the answer (i.e. $\text{Sufficiency}((s_i)) \geq \tau$);
- it contains a boxed answer expression (i.e. "`\boxed{...}`") whose value is equivalent to the ground-truth answer;
- it contains a direct answer statement (a phrase beginning with one of "`Thus`", "`Therefore`", "`So`", "`Hence`", "`Consequently`", "`This means`", "`My`", "`The`"; contains the word "`answer`" or "`result`"; and some answer value).

B.2 Identifying a core reasoning subsequence

In this section we present the full details on the different variants to identify the core reasoning subsequence ($\mathcal{T}^*$), as presented in Section 3.

B.2.1 Activation-based variant

We present the full code for the activation-based variant. Function 1 implements the first stage, building the sequence of steps with direct or indirect influence on the answer ($\mathcal{T}_{\text{inf}}$). Steps excluded from $\mathcal{T}_{\text{inf}}$ are labeled as removable for later classification experiments.

Function 2 implements the second stage, building the core reasoning subsequence $\mathcal{T}^*$. Steps included in $\mathcal{T}^*$ are labeled as important and non-removable.

```python
def get_influence_mask(influence_matrix, threshold):
  """
  Args:
  - influence_matrix: A (p x p) matrix where entry (j, i) represents the
      influence of step i on step j.
  - threshold: A float between 0 and 1 representing the fraction of total
      influence to retain for each step.
  """
  p = influence_matrix.shape[0]
  step_mask = torch.zeros(p, dtype=torch.long)
  step_mask[p - 1] = 1
  for ref_idx in range(p - 1, 0, -1):
    if step_mask[ref_idx] != 0:
      influence_on_step = influence_matrix[ref_idx, :]

      # Sort in descending order
      sorted_inf = torch.sort(influence_on_step, descending=True).values

      # Accumulate and Normalize
      cumulative_score = torch.cumsum(sorted_inf, dim=0) / torch.sum(sorted_inf)

      # Find score threshold that keeps the desired fraction of total influence
      threshold_index = torch.searchsorted(cumulative_score, threshold).item()
      threshold_influence_value = sorted_inf[threshold_index]

      # Include all steps that contribute to the influence in the mask
      step_mask |= (influence_on_step >= threshold_influence_value)

  return step_mask
```

Listing 1: Activation-based variant function to build $\mathcal{T}_{\text{inf}}$.

```python
def get_core_reasoning_subsequence(influence_matrix):
  """
  Args:
  - influence_matrix: A (p x p) matrix where entry (j, i) represents the
      influence of step i on step j.
  """
  THRESHOLDS = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0]
  for t in THRESHOLDS:
    mask = get_influence_mask(influence_matrix, t)

    suff, atnb = evaluate_reasoning_subsequence(mask)
    # Stop if sufficient mask is found or all steps are kept
    if suff >= SUFFICIENCY_THRESHOLD and atnb >= ATTAINABILITY_THRESHOLD:
      break

  # Calculate average influence
  mean_influence = influence_matrix.sum(dim=0) / (influence_matrix != 0).sum(dim=0)

  # Get ranking by average influence
  ranked_step_indices = mean_influence.argsort()

  # Only consider steps that were kept in the last threshold result
  # Build queue sorted by ascending mean influence
  queue = [idx for idx in mean_influence.argsort()
           if mask[idx] == 1 and idx != len(mask) - 1]
  failures_since_last_removal = 0

  while queue and failures_since_last_removal < len(queue):
    step_idx = queue.pop(0)
    eval_mask = mask.clone()
    eval_mask[step_idx] = 0
    suff, atnb = evaluate_reasoning_subsequence(eval_mask)
    if suff >= SUFFICIENCY_THRESHOLD and atnb >= ATTAINABILITY_THRESHOLD:
      # Confirm removal
      mask = eval_mask
      failures_since_last_removal = 0
    else:
      # Revert and re-queue at end
      queue.append(step_idx)
      failures_since_last_removal += 1

  core_subsequence = [step_idx for step_idx in range(len(mask)) if mask[step_idx] == 1]
  return core_subsequence
```

Listing 2: Activation-based variant function to build $\mathcal{T}^*$.

B.2.2 Token-based variant

For the token-based variant, we use Gemini-2.5-Pro (Comanici et al., 2025) as the external LLM for both stages of the greedy procedure ($\mathcal{T}_p \to \mathcal{T}_{\text{inf}}$ and $\mathcal{T}_{\text{inf}} \to \mathcal{T}^*$). In both stages, Gemini is called through the Gemini API, using greedy decoding (temperature $t = 0$) with up to $24k$ output tokens. We approximate the total cost of API calls for this at 50 USD.

In the first stage, we prompt Gemini to identify reasoning steps that do not contribute to the final answer (Prompt 3); steps excluded from the result are labeled as removable. We then prompt Gemini for a ranking of the remaining steps by importance (Prompt 4). This ranking drives the iterative removal of the least important steps (similarly to Listing 2), and results in the core reasoning subsequence $\mathcal{T}^*$.

You are analyzing a mathematical chain-of-thought (CoT) reasoning trace.
You are given the input question and the minimal reasoning trace required to answer, split to steps.
Your task is to identify which reasoning steps do NOT contribute directly or indirectly to producing the final answer.
A step contributes directly if it is used in reaching the final answer.
A step contributes indirectly if it influences another step that recursively affects the final answer.
A step does NOT contribute if removing it would not affect the reasoning chain's ability to reach the correct answer — for example, steps that are redundant, restatements, or tangential.
Don't consider the last sentence (the answer step).
The question: <question>{question}</question>
The reasoning steps:
{sentence_lines}
After thinking, output your final answer as a JSON list of step indices that do NOT contribute to the answer and can be safely removed. Output the list inside a "'json code block, like:
"'json
[removable_idx_1, removable_idx_2, ...]
"'

Listing 3: Gemini prompt to identify $\mathcal{T}_{\text{inf}}$ in the token-based variant.

You are analyzing a mathematical chain-of-thought (CoT) reasoning trace.
You are given the input question and the minimal reasoning trace required to answer, split to steps.
Your task is to rank ALL reasoning steps by their importance in producing the final answer.
A step is important if it directly contributes to the final answer, or if it indirectly influences another step that recursively affects the final answer. A step is unimportant if removing it would not affect the reasoning chain's ability to reach the correct answer.
Think about which steps are load-bearing for the argument and which are redundant, restatements, or tangential.
Don't rank the last sentence.
The question: <question>{question}</question>
The reasoning steps:
{sentence_lines}
After thinking, output your final ranking as a JSON list of step indices ordered from MOST important to LEAST important. The list must contain exactly the indices {valid_indices}.
Output the ranking inside a "'json code block, like:
"'json
[most_important_idx, ..., least_important_idx]
"'

Listing 4: Gemini prompt to rank $\mathcal{T}_{\text{inf}}$ steps by importance for the token-based variant.

B.3 Classifying reasoning steps

In this section we present the full details on methods for importance classification (Section 4).

B.3.1 Activation-based probe importance classification

In this section we describe the technical details regarding the activation-based importance probes.

Probe architectures and input representations. We evaluate four probe families:

1. Linear: a single linear layer from the averaged activation vector to a scalar.

2. MLP: an MLP with a ReLU non-linearity. We sweep over hidden dimensions $128, 256$ and depth $2, 5$.
3. Layerwise-Linear (LW-Linear): We flatten the (L, D) per-layer activation representation and apply a single linear layer.
4. Layerwise-MLP (LW-MLP): the same flattened input is passed through a 2-layer MLP. We sweep over the hidden probe dimension from $16, 128, 1024, 4096$.

In the first two architectures (Linear and MLP), the activation tensors are averaged across both token and layer dimensions, resulting in $d-$shaped inputs, where d is the model's hidden dimension. Across the latter architectures (LW-Linear and LW-MLP), the activations are averaged only across the token dimension, resulting in a $(L, d)-$shaped inputs. We additionally tested shallow transformer encoders (1–2 layers) operating on the full $(S, L, d)-$shaped activation tensors, but these underperformed the simpler probes while incurring substantially higher training costs.

Training. Probes are trained for 30 epochs with AdamW (with learning rate 3×10^{-4}, weight decay 10^{-4}, batch size 32 and cosine annealing learning rate scheduler) using binary cross-entropy loss. The labeled reasoning steps are split to an $80/20$ training/evaluation split, and repeat the process across 3 seeds. We report the best evaluation accuracy achieved across epochs.

Results. The results across all tested settings are presented in Tables 2 and 3 for the HARP and MATH-500 datasets, respectively. Across most model–dataset combinations, the layerwise-MLP probe with hidden dimension 128 performs best, motivating our choice of it as the standard probe in Sections 4.1 and 5.

Table 2: Probe architecture search results on HARP

Configuration	GPT-OSS	DS-1.5	DS-7	DS-14	Olmo3
Context, All Layers, Linear	$73.0\%_{\pm 0.6\%}$	$75.6\%_{\pm 1.0\%}$	$77.8\%_{\pm 1.0\%}$	$77.6\%_{\pm 0.3\%}$	$75.0\%_{\pm 0.6\%}$
Context, All Layers, MLP (h=128, L=2)	$79.2\%_{\pm 0.1\%}$	$78.8\%_{\pm 1.1\%}$	$81.5\%_{\pm 0.5\%}$	$80.3\%_{\pm 0.2\%}$	$80.3\%_{\pm 0.2\%}$
Context, All Layers, MLP (h=128, L=5)	$78.6\%_{\pm 0.4\%}$	$78.3\%_{\pm 1.3\%}$	$81.3\%_{\pm 0.9\%}$	$80.6\%_{\pm 0.7\%}$	$79.6\%_{\pm 0.3\%}$
Context, All Layers, MLP (h=256, L=2)	$79.1\%_{\pm 0.3\%}$	$79.4\%_{\pm 1.6\%}$	$82.0\%_{\pm 1.1\%}$	$81.0\%_{\pm 0.3\%}$	$80.2\%_{\pm 0.4\%}$
Context, All Layers, MLP (h=256, L=5)	$79.1\%_{\pm 0.2\%}$	$79.4\%_{\pm 1.7\%}$	$81.8\%_{\pm 1.0\%}$	$80.9\%_{\pm 0.4\%}$	$80.1\%_{\pm 0.2\%}$
Context, All Layers, LW-Linear	$75.0\%_{\pm 0.4\%}$	$77.6\%_{\pm 1.0\%}$	$80.4\%_{\pm 0.7\%}$	$79.7\%_{\pm 0.2\%}$	$79.2\%_{\pm 0.3\%}$
Context, All Layers, LW-MLP (h=16)	$78.5\%_{\pm 0.3\%}$	$79.5\%_{\pm 1.3\%}$	$82.4\%_{\pm 0.3\%}$	$81.9\%_{\pm 0.5\%}$	$80.5\%_{\pm 0.6\%}$
Context, All Layers, LW-MLP (h=128)	$\mathbf{80.0\%}_{\pm 0.2\%}$	$\mathbf{80.6\%}_{\pm 1.1\%}$	$82.5\%_{\pm 0.2\%}$	$82.5\%_{\pm 0.3\%}$	$\mathbf{81.6\%}_{\pm 0.6\%}$
Context, All Layers, LW-MLP (h=1024)	$\mathbf{80.0\%}_{\pm 0.3\%}$	$80.3\%_{\pm 1.2\%}$	$\mathbf{82.6\%}_{\pm 0.3\%}$	$82.4\%_{\pm 0.4\%}$	$81.1\%_{\pm 0.8\%}$
Context, All Layers, LW-MLP (h=4096)	$80.0\%_{\pm 0.1\%}$	$80.0\%_{\pm 1.2\%}$	$82.4\%_{\pm 0.2\%}$	$\mathbf{82.6\%}_{\pm 0.2\%}$	$81.5\%_{\pm 0.6\%}$
No Context, All Layers, Linear	$66.6\%_{\pm 0.2\%}$	$56.7\%_{\pm 0.8\%}$	$57.6\%_{\pm 1.1\%}$	$71.0\%_{\pm 0.7\%}$	$70.4\%_{\pm 0.6\%}$
No Context, All Layers, MLP (h=128, L=2)	$69.2\%_{\pm 0.2\%}$	$56.8\%_{\pm 0.9\%}$	$57.9\%_{\pm 0.8\%}$	$72.4\%_{\pm 1.1\%}$	$70.4\%_{\pm 0.2\%}$
No Context, All Layers, MLP (h=128, L=5)	$69.4\%_{\pm 0.4\%}$	$56.9\%_{\pm 0.9\%}$	$56.9\%_{\pm 0.6\%}$	$71.3\%_{\pm 0.0\%}$	$71.3\%_{\pm 0.4\%}$
No Context, All Layers, MLP (h=256, L=2)	$69.0\%_{\pm 0.0\%}$	$56.7\%_{\pm 0.9\%}$	$56.9\%_{\pm 0.5\%}$	$72.9\%_{\pm 0.8\%}$	$70.4\%_{\pm 0.1\%}$
No Context, All Layers, MLP (h=256, L=5)	$68.8\%_{\pm 0.4\%}$	$56.7\%_{\pm 0.9\%}$	$57.7\%_{\pm 0.1\%}$	$71.8\%_{\pm 0.6\%}$	$70.7\%_{\pm 0.1\%}$
No Context, All Layers, LW-Linear	$65.5\%_{\pm 0.2\%}$	$56.9\%_{\pm 0.8\%}$	$56.7\%_{\pm 0.2\%}$	$71.0\%_{\pm 0.5\%}$	$71.1\%_{\pm 0.4\%}$
No Context, All Layers, LW-MLP (h=16)	$69.0\%_{\pm 0.5\%}$	$56.7\%_{\pm 0.9\%}$	$58.3\%_{\pm 0.7\%}$	$72.5\%_{\pm 0.4\%}$	$70.7\%_{\pm 0.4\%}$
No Context, All Layers, LW-MLP (h=128)	$68.9\%_{\pm 0.3\%}$	$56.7\%_{\pm 0.9\%}$	$57.3\%_{\pm 0.1\%}$	$72.7\%_{\pm 0.5\%}$	$70.5\%_{\pm 0.6\%}$
No Context, All Layers, LW-MLP (h=1024)	$69.6\%_{\pm 0.2\%}$	$56.7\%_{\pm 0.9\%}$	$57.7\%_{\pm 0.6\%}$	$71.4\%_{\pm 0.2\%}$	$70.9\%_{\pm 0.2\%}$
No Context, All Layers, LW-MLP (h=4096)	$69.6\%_{\pm 0.3\%}$	$56.7\%_{\pm 0.9\%}$	$57.7\%_{\pm 1.0\%}$	$71.9\%_{\pm 0.3\%}$	$71.4\%_{\pm 0.3\%}$

B.3.2 Baselines for importance classification

In this section we describe the technical details regarding the baselines for importance detection (Section 4.2).

LLM-as-a-judge. We prompt Gemini-2.5-Pro (Comanici et al., 2025) with a classification prompt containing a target reasoning step, asking it to determine whether the step is removable or non-removable. We use greedy decoding (temperature $t = 0$) via the Gemini API, allowing up to $24k$ output tokens. The number of labeled reasoning steps can reach tens of thousands of entries, making classification prohibitively expensive. Thus, we sample 1500 steps per model per dataset (while maintaining a balanced split between removable and non-removable labels). We approximate the total cost of API calls for this at 150 USD. The classification prompts are provided in Prompts 5 and 6 for classification without context and with full context, respectively.

Table 3: Probe architecture search results on MATH-500

Configuration	GPT-OSS	DS-1.5	DS-7	DS-14	Olmo3
Context, All Layers, Linear	$78.0\%_{\pm 1.2\%}$	$75.8\%_{\pm 3.8\%}$	$82.9\%_{\pm 0.9\%}$	$74.0\%_{\pm 3.1\%}$	$78.3\%_{\pm 0.5\%}$
Context, All Layers, MLP (h=128, L=2)	$83.4\%_{\pm 1.3\%}$	$77.8\%_{\pm 1.8\%}$	$\mathbf{86.1\%}_{\pm 1.5\%}$	$79.2\%_{\pm 1.7\%}$	$82.0\%_{\pm 1.5\%}$
Context, All Layers, MLP (h=128, L=5)	$82.3\%_{\pm 1.5\%}$	$72.6\%_{\pm 3.4\%}$	$85.9\%_{\pm 1.3\%}$	$71.9\%_{\pm 2.3\%}$	$82.4\%_{\pm 0.8\%}$
Context, All Layers, MLP (h=256, L=2)	$83.3\%_{\pm 0.9\%}$	$78.3\%_{\pm 1.0\%}$	$86.0\%_{\pm 1.8\%}$	$78.5\%_{\pm 1.3\%}$	$82.3\%_{\pm 1.9\%}$
Context, All Layers, MLP (h=256, L=5)	$83.3\%_{\pm 1.0\%}$	$77.1\%_{\pm 1.5\%}$	$84.8\%_{\pm 1.6\%}$	$75.0\%_{\pm 2.3\%}$	$\mathbf{82.8\%}_{\pm 1.3\%}$
Context, All Layers, LW-Linear	$80.9\%_{\pm 1.5\%}$	$80.5\%_{\pm 3.7\%}$	$85.6\%_{\pm 1.7\%}$	$79.5\%_{\pm 2.7\%}$	$80.5\%_{\pm 0.9\%}$
Context, All Layers, LW-MLP (h=16)	$83.4\%_{\pm 1.7\%}$	$\mathbf{81.3\%}_{\pm 0.6\%}$	$82.3\%_{\pm 2.5\%}$	$78.5\%_{\pm 2.7\%}$	$82.3\%_{\pm 1.7\%}$
Context, All Layers, LW-MLP (h=128)	$83.8\%_{\pm 1.0\%}$	$80.5\%_{\pm 0.9\%}$	$83.5\%_{\pm 1.9\%}$	$\mathbf{80.5\%}_{\pm 2.2\%}$	$\mathbf{82.8\%}_{\pm 1.9\%}$
Context, All Layers, LW-MLP (h=1024)	$83.4\%_{\pm 1.1\%}$	$80.4\%_{\pm 1.6\%}$	$83.0\%_{\pm 1.8\%}$	$79.9\%_{\pm 4.8\%}$	$\mathbf{82.8\%}_{\pm 1.8\%}$
Context, All Layers, LW-MLP (h=4096)	$\mathbf{84.2\%}_{\pm 0.4\%}$	$79.4\%_{\pm 1.0\%}$	$83.1\%_{\pm 2.8\%}$	$77.8\%_{\pm 5.8\%}$	$82.2\%_{\pm 1.6\%}$
No Context, All Layers, Linear	$72.0\%_{\pm 0.4\%}$	$58.5\%_{\pm 2.1\%}$	$58.5\%_{\pm 2.1\%}$	$58.0\%_{\pm 2.7\%}$	$74.5\%_{\pm 1.5\%}$
No Context, All Layers, MLP (h=128, L=2)	$75.2\%_{\pm 0.0\%}$	$60.5\%_{\pm 1.5\%}$	$58.9\%_{\pm 1.0\%}$	$61.1\%_{\pm 1.3\%}$	$73.6\%_{\pm 1.0\%}$
No Context, All Layers, MLP (h=128, L=5)	$73.7\%_{\pm 1.5\%}$	$60.5\%_{\pm 2.4\%}$	$59.6\%_{\pm 1.7\%}$	$56.9\%_{\pm 3.5\%}$	$73.6\%_{\pm 0.3\%}$
No Context, All Layers, MLP (h=256, L=2)	$75.5\%_{\pm 0.9\%}$	$59.4\%_{\pm 0.8\%}$	$59.0\%_{\pm 1.4\%}$	$61.1\%_{\pm 2.1\%}$	$73.6\%_{\pm 0.5\%}$
No Context, All Layers, MLP (h=256, L=5)	$75.2\%_{\pm 0.7\%}$	$59.4\%_{\pm 1.6\%}$	$57.3\%_{\pm 2.0\%}$	$59.7\%_{\pm 4.0\%}$	$73.4\%_{\pm 0.3\%}$
No Context, All Layers, LW-Linear	$73.4\%_{\pm 0.6\%}$	$59.7\%_{\pm 2.6\%}$	$58.9\%_{\pm 0.7\%}$	$59.0\%_{\pm 4.7\%}$	$73.0\%_{\pm 0.5\%}$
No Context, All Layers, LW-MLP (h=16)	$75.8\%_{\pm 1.1\%}$	$59.4\%_{\pm 1.0\%}$	$59.6\%_{\pm 0.9\%}$	$59.4\%_{\pm 1.5\%}$	$72.6\%_{\pm 1.4\%}$
No Context, All Layers, LW-MLP (h=128)	$75.6\%_{\pm 2.0\%}$	$57.6\%_{\pm 2.6\%}$	$60.3\%_{\pm 1.1\%}$	$61.5\%_{\pm 1.5\%}$	$72.3\%_{\pm 0.6\%}$
No Context, All Layers, LW-MLP (h=1024)	$74.0\%_{\pm 1.5\%}$	$59.3\%_{\pm 1.9\%}$	$59.7\%_{\pm 2.0\%}$	$61.8\%_{\pm 3.4\%}$	$74.3\%_{\pm 0.4\%}$
No Context, All Layers, LW-MLP (h=4096)	$75.3\%_{\pm 0.4\%}$	$59.4\%_{\pm 1.0\%}$	$57.6\%_{\pm 1.5\%}$	$61.1\%_{\pm 1.3\%}$	$74.1\%_{\pm 0.4\%}$

TF-IDF representation baseline. We represent each reasoning step as a sparse TF-IDF vector using scikit-learn's (Pedregosa et al., 2011) `TfidfVectorizer` with sublinear term-frequency scaling. We use n-grams of $n \in [1, 5]$, and a vocabulary capped at $100{,}000$ features. We use a two-layer MLP with an input dimension of $100,000$, a hidden dimension of 128 and a ReLU non-linearity. We train it for 30 epochs with AdamW (with learning rate 1×10^{-3}, weight decay 10^{-4}, batch size 256) using binary cross-entropy loss.

Sentence transformer representation baseline. We use `all-MiniLM-L6-v2` as the pre-trained sentence transformer. We use the same classifier and training scheme as the TF-IDF baseline.

```
You are labeling a sentence from a mathematical chain-of-thought (CoT) reasoning trace.
Your task is to predict whether this sentence is **removable** or **nonremovable**.
Definitions:
- **removable**: The sentence can be deleted from the trace such that all subsequent
sentences remain logically supported and the final answer is still reachable from the remaining
sentences alone.
- **nonremovable**: The sentence is essential to the reasoning chain. Without it, at least one
subsequent sentence loses its logical basis, or the final answer can no longer be derived from
the remaining sentences.
To make your judgment, consider whether the sentence:
- Introduces a new variable, equation, or fact that is used by a later sentence.
- Performs a calculation or derivation whose result is referenced downstream.
- Or instead restates the problem, expresses uncertainty, narrates the approach, summarizes
prior steps, or makes a tangential observation.
Based only on the sentence text, predict whether this type of sentence is typically removable
or nonremovable in a reasoning trace.
Think step by step about what role this sentence most likely plays, then on the last line output
ONLY the word "removable" or "nonremovable".
Sentence: {sentence}
```

Listing 5: Prompt to classify a reasoning step as removable or non-removable, without context.

You are labeling a sentence from a mathematical chain-of-thought (CoT) reasoning trace.
Your task is to predict whether this sentence is **removable** or **nonremovable**.
Definitions:
- **removable**: The sentence can be deleted from the trace such that all subsequent sentences remain logically supported and the final answer is still reachable from the remaining sentences alone.
- **nonremovable**: The sentence is essential to the reasoning chain. Without it, at least one subsequent sentence loses its logical basis, or the final answer can no longer be derived from the remaining sentences.
To make your judgment, consider whether the sentence:
- Introduces a new variable, equation, or fact that is used by a later sentence.
- Performs a calculation or derivation whose result is referenced downstream.
- Or instead restates the problem, expresses uncertainty, narrates the approach, summarizes prior steps, or makes a tangential observation.
You are given the input question and the full CoT reasoning trace with sentence indices. The target sentence to label is marked with »>. Read the full chain of thought, then assess whether the marked sentence's content is used or referenced by any subsequent sentence, or whether it provides information necessary to reach the final answer.
Think step by step about the sentence's role in the chain, then on the last line output ONLY the word "removable" or "nonremovable".
The question: `<question>`{question}`</question>`
The CoT sentences (the target sentence is marked with »>):
[0] `FIRST STEP`
[1] `SECOND STEP`
>>> [2] `TARGET STEP`
[3] `FOURTH STEP`
[4] ...

Listing 6: Prompt to classify a reasoning step as removable or non-removable, with full context.

B.4 Correlation of probes with surface-level features

The surface-level analysis (in Section 5.2) tests whether the performance of activation-based importance probes can be matched by shallow, surface-level textual cues. We extract the following surface-level features for each reasoning step:

- **Relative position**: The relative position of the reasoning step, normalized by the number of reasoning steps (for step s_i out of n steps, this is equal to $\frac{i}{n}$.
- **Token length**: The number of tokens within the reasoning step.
- **Numbers count**: Count of separate numeric literals in the reasoning step.
- **Arithmetic operator count**: Count of arithmetic and relational symbols in $\{+, -, *, /, ', =, <, >, \leq, \geq, \neq, \pm\}$.

We also extract a set of categorical features, where is reasoning step is assigned a binary label for each of the following categories, set to 1 if the step contains any of the category's phrases (matches case-insensitively). The categories and their corresponding phrases are:

- **Filler**: "let me think", "let's think", "lets think", "let me consider", "let's consider", "let me see", "let's see", "let me ensure", "let's ensure", "lets ensure", "hmm", "okay", "alright", "so basically", "in other words".
- **Verification**: "wait", "double check", "double-check", "recompute", "recalculate", "verify", "let me check", "let's check", "checking", "is this correct", "does this make sense", "let me re", "actually, let me", "hold on".
- **Planning**: "we need", "first, i", "step 1", "step 2", "my approach", "i'll start by", "the plan is", "i need to", "i should", "let me start", "to solve this", "the strategy", "i will", "my strategy".

- **Computation**: “calculating”, “computing”, “plugging in”, “substituting”, “evaluating”, “simplifying”, “expanding”, “factoring”, “multiplying”, “dividing”, “adding”, “subtracting”, “integrating”, “differentiating”, “solving for”.
- **Fact retrieval**: “we know that”, “recall that”, “remember that”, “by definition”, “by the formula”, “using the formula”, “the formula for”, “it is known”, “a known result”, “from the theorem”, “by theorem”, “since we know”.

C Sufficiency and attainability thresholds

We discuss our choice of sufficiency and attainability thresholds, and present supporting ablations. In the main paper, both metrics share a notation $\tau = 0.8$; for clarity we give a separate notation ($\tau_{\text{suff}}, \tau_{\text{atnb}}$) for this appendix.

Attainability threshold. Attainability is a continuous measure that controls how far the core reasoning subset may deviate from the model's distribution when reasoning steps are removed. A higher threshold restricts removal more strictly, keeping the remaining chain closer to in-distribution outputs. Thus, to compare different values of τ_{atnb}, we measure how the choice of threshold affects the prefix length $|\mathcal{T}_p|$ and the amount of removable and non-removable steps, as described in Section 3.

We ablate over $\tau_{\text{atnb}} \in \{0.6, 0.8, 0.9, 0.99\}$. Values below 0.6 lead to degenerate outputs, making attainability ineffective, so we ignore them. Due to computational cost, we perform this analysis only on the Math-500 dataset.

For each threshold, we recompute the sufficient prefixes and rerun the activation-based variant (Section 3) on all prompts. For each prompt, we record three quantities, each expressed as a percentage of the full reasoning chain: the prefix length in steps, the fraction of steps labeled removable, and the fraction labeled non-removable (i.e., the core reasoning subsequence size). We average each quantity across prompts (per threshold and model) and report the results in Figure 5

The results are consistent across models. The prefix length (blue) stays nearly constant across thresholds, rising only at $\tau_{\text{atnb}} = 0.99$. The core reasoning subset size (orange, non-removable steps) follows the same pattern. This is expected, as a stricter threshold limits how many steps the method can remove (green) before the chain goes out of distribution, which inflates the core subset with steps that are not strictly essential.

Due to the similar performance of the three analyzed thresholds $\tau_{\text{atnb}} \in \{0.6, 0.8, 0.9\}$, we default to choosing the middle $\tau_{atnb} = 0.8$. We opt to not use the more strict $\tau_{atnb} = 0.99$ as it leads larger core reasoning subsets, that contain non-essential steps, which remain only to better match the model's output distribution. We do note, however, that this is a tradeoff: applications that require a more in-distribution core reasoning chain could use the higher $\tau_{\text{atnb}} = 0.99$ threshold.

Sufficiency threshold. Sufficiency is measured as the fraction of correct answer out of $K = 5$ completions. Thus, the set of relevant thresholds is $\tau_{\text{suff}} \in \{0.2, 0.4, 0.6, 0.8, 1.0\}$, each corresponding to $i \in [1, K]$ correct completions. To focus on reasoning chains where the model answers correctly with high confidence, we only consider $\tau_{\text{suff}} \in \{0.8, 1.0\}$. We choose $\tau_{\text{suff}} = 0.8$ as the sufficiency threshold for our analysis, corresponding to $i = 4$ correct answer out of 5 completions, to tolerate occasional sampling errors while still requiring strong overall correctness.

D Performance statistics

We report, for each model-dataset pair, the number of prompts that pass successive filtering stages in our analysis. Specifically, we report five prompt counts per setting: (i) the total number of analyzed prompts; (ii) the number of prompts for which the model completed the generation of the reasoning chain (i.e. generation did not terminate due to token limit); (iii) the number of prompts for which the reasoning chain is necessary; (iv) the number of prompts for which the model produced a correct answer; and (v) the number of prompts for which the reasoning chain remains sufficient to obtain correct answer even after the removal of shortcut steps. The full results are reported in Table 4.

We also report, for each model-dataset pair, several step-level statistics in Table 6 and appendix D. We report several metrics: (i) average number of steps in each reasoning chain; (ii) the average number

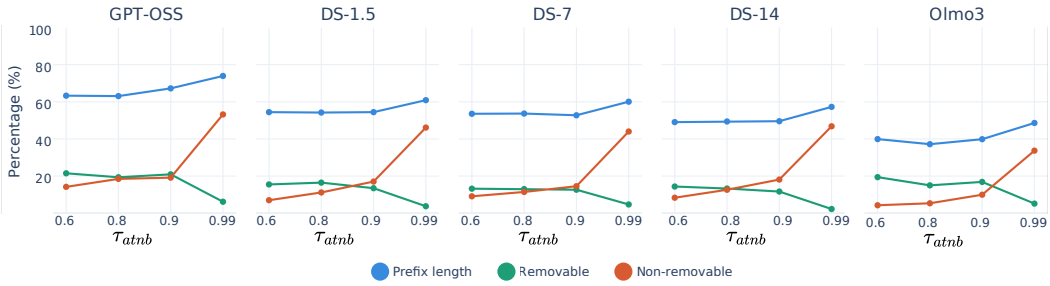


Figure 5: **The effect of different attainability thresholds (τ_{atnb}) on the resulting lengths of removal experiments.**

Table 4: **Number of prompts after each filtering, across models and datasets.**

		GPT-OSS	DS-1.5	DS-7	DS-14	Olmo-3
HARP	Total	5280	4886	4886	5280	4886
	Reasoning Complete	4726	2752	3825	4019	3194
	Reasoning Necessary	4360	2465	3219	3391	2838
	Correct Answer	3578	2169	2808	3014	2539
	Shortcut-removed Suff.	2430	726	938	1022	1087
Math-500	Total	500	500	500	500	500
	Reasoning Complete	479	370	437	453	408
	Reasoning Necessary	397	295	314	339	324
	Correct Answer	330	262	279	291	299
	Shortcut-removed Suff.	235	114	130	121	156

of steps in the sufficient prefix, as a percentage of the total length; (iii-iv) the average number of steps labeled as removable and non-removable, as a percentage of the sufficient prefix length.

Table 5: **Reasoning step statistics (MATH-500 dataset).** Prefix length is shown as percentage of total steps. Removable and non-removable are shown as percentage of prefix steps.

Table 6: **Reasoning step statistics (HARP dataset).** Prefix length is shown as percentage of total steps. Removable and non-removable are shown as percentage of prefix steps.

	GPT-OSS	**DS-1.5**	**DS-7**	**DS-14**	**Olmo3**
Avg. Steps	96.5 ± 75.8	95.7 ± 53.1	99.4 ± 57.3	109.3 ± 68.3	191.4 ± 90.2
Prefix Length	$64.2\% \pm 21.8$	$56.1\% \pm 21.1$	$51.2\% \pm 22.3$	$55.2\% \pm 23.0$	$34.1\% \pm 18.0$
Removable	$37.6\% \pm 26.8$	$27.3\% \pm 21.0$	$28.1\% \pm 21.9$	$30.6\% \pm 23.1$	$36.4\% \pm 25.6$
Non-removable	$22.3\% \pm 15.7$	$24.4\% \pm 16.2$	$23.4\% \pm 18.0$	$25.4\% \pm 18.3$	$20.1\% \pm 13.8$

	GPT-OSS	**DS-1.5**	**DS-7**	**DS-14**	**Olmo3**
Avg. Steps	68.2 ± 66.6	79.8 ± 46.4	86.7 ± 52.3	87.3 ± 57.9	170.4 ± 83.2
Prefix Length	$63.1\% \pm 22.2$	$54.2\% \pm 22.6$	$53.7\% \pm 21.0$	$49.4\% \pm 21.0$	$37.2\% \pm 19.0$
Removable	$29.6\% \pm 24.1$	$30.2\% \pm 21.5$	$24.0\% \pm 21.2$	$26.7\% \pm 22.6$	$38.5\% \pm 26.0$
Non-removable	$30.3\% \pm 16.9$	$22.1\% \pm 13.0$	$23.3\% \pm 17.3$	$27.3\% \pm 15.0$	$17.4\% \pm 11.3$

E Additional results

E.1 Different input attribution methods

In the activation-based variant (Section 3) we employ simple gradient attribution to measure influence between reasoning steps. To ablate the effect of this choice on the importance labels assigned to reasoning steps, we compare it against alternative activation-based attribution methods. Specifically, we compute $\mathbf{I}(s_i, s_j)$ using either gradient-input attribution (Shrikumar et al., 2017) or integrated gradients (Sundararajan et al., 2017) with $n = 5$ accumulation steps. The baseline used for integrated gradients is the average of all tokens in the reasoning chain. We evaluate each method by measuring the size of the core reasoning subsequence as a percentage of the sufficient prefix (identically to Table 1), and average across prompts.

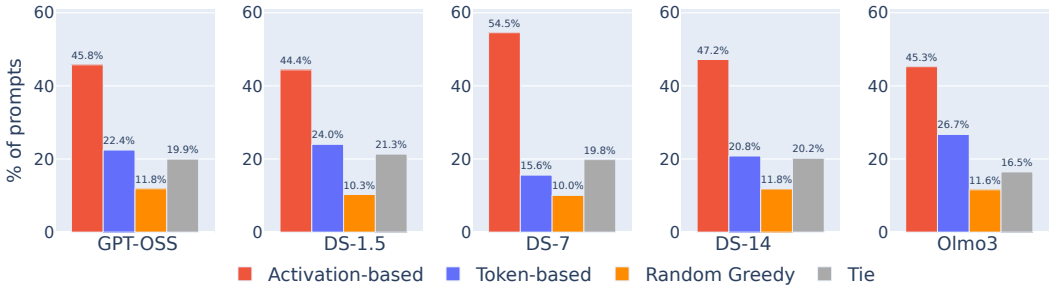
Results on the Math-500 dataset are reported in Table 7. All three attribution methods yield similar results, each outperforming both the token-based variant and the randomized variant in Section 3. While the integrated gradients method yields marginally smaller core reasoning subsequences, its improvement is small and comes at a substantially higher computational cost. We thus adopt the similarly-performing gradient-only scores across our remaining experiments.

Table 7: **Comparing the size of the resulting core reasoning subset using different gradient attribution methods.**

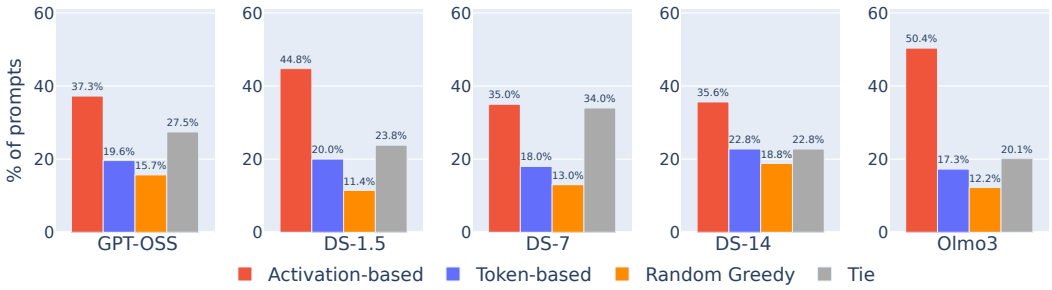
Method	GPT-OSS	DS-1.5	DS-7	DS-14	Olmo3
Gradient	$30.8\% \pm 13.8$	$22.1\% \pm 13.0$	$23.1\% \pm 17.2$	$27.8\% \pm 13.6$	$18.0\% \pm 11.2$
Gradient $\times$ Input	$26.9\% \pm 13.9$	$22.2\% \pm 13.2$	$21.0\% \pm 18.0$	$29.2\% \pm 14.8$	$19.7\% \pm 16.2$
Integrated Gradients	$24.3\% \pm 11.4$	$20.5\% \pm 11.1$	$19.2\% \pm 14.7$	$26.1\% \pm 14.8$	$18.1\% \pm 12.3$

E.2 Per-prompt comparison of core subsequence identification

In addition to reporting the average size of core reasoning subsequences $\mathcal{T}^*$ identified by each variant in Table 1, we compare the three variants per-prompt. For each prompt, we check which variant yields the most $\mathcal{T}^*$. We report the percentage of prompts for which each variant wins, along with the percentage of ties (Figure 6). Across models and datasets, the activation-based variant, which utilizes gradient influence as an internal signal, yields the best results on the largest share of prompts. This is further motivation for our choice to use it to identify core reasoning subsets.



(a) **Per-prompt comparison of the three variants on HARP.** We report the percentage of prompts for which each variant identifies the most concise core reasoning subset.



(b) **Per-prompt comparison of the three variants on MATH-500.** We report the percentage of prompts for which each variant identifies the most concise core reasoning subset.

Figure 6

E.3 Reasoning step examples and labels

We present qualitative examples of importance labels for individual reasoning steps. These examples are organized into 4 groups, defined by the combination of their ground-truth label (removable or non-removable) and activation probe prediction success (Section 4.1). Each group is colored as light red if the steps within it are removable, or light green if they are important and non-removable. The title of each group indicates if the full-context activation probes correctly classified the group's steps.

The steps are *randomly sampled* from the GPT-OSS-20B analysis on MATH-500. Full results, including all labeled steps in all experimental settings, will be made publicly available.

These examples illustrate that surface-level features alone are often insufficient for predicting importance: steps of the same surface type—fillers, direct computations, and others—can appear with different ground truth importance labels.

GT: Removable | Activation Probe Correct: ✓

1. Actually parallelogram sides: one side is vertical segment between (0,c) and (0,d).
2. Let's analyze.
3. Let's analyze region.
4. So sign of A and B?
5. We can write as $e^{i\theta}(e^{-i\theta} - 1 + e^{i\theta}) = e^{i\theta}(1 - e^{i\theta} + e^{i2\theta})$?
6. Term1: $(x - z)^2/2 = (x^2 - 2xz + z^2)/2$. Good.
7. But we must check: Did we compute correctly?
8. Solve congruences.
9. So AC direction vector (x_C, h) is bisector of angle between BA and AD.
10. Wait sign: -4xy?
11. 13/9 = 52/36.
12. Similarly odd: $\Pi_{k=1}^{n}((2k-1)^2 + (2k-1) + 1/2)((2k-1)^2 - (2k-1) + 1/2)$.
13. So we have factorization: $x^4 + 1/4 = (x^2 + x + 1/2)(x^2 - x + 1/2)$. Good.
14. So include.
15. So we need to find intersection with AC: G + t(c-b) = u c.

GT: Removable | Activation Probe Correct: ×

1. So x must be $\geq (3 + sqrt33)/2 \approx 4.372$.
2. Because sum of 0 to 26 = 26*27/2 = 351.
3. Without loss of generality, we can choose F to be a particular face, say 1?
4. We can solve.
5. But we should also check interior points maybe produce lower?
6. But we need actual integer values.
7. r^2 must be one of these.
8. So square: $4(3x + 6) = 4(3x + 6)$. trivial.
9. So compare coefficients:
10. So $\approx 0.4097 < 1$. Good.
11. Then compute their average.
12. a/(b+c) = a/(c(v+1)) = (a/c)/(v+1) = u/(v+1).
13. But f is not one-to-one globally.
14. Area = $\int_{x=0}^{1}[(x+1)-(1-x)]dx + \int_{x=1}^{2}[(x+1)-(x-1)]dx + \int_{x=2}^{3}[3-(x-1)]dx$.
15. Then $u^2 = v^2 - 9 = 36 \pm 4sqrt41$.

GT: Non-removable | Activation Probe Correct: ✓

1. So equation: $6r^2 - 19r - 7 = 8r^2 - 34r + 21$.
2. So compute: $(3 - (-4 - 5i)) = 3 + 4 + 5i = 7 + 5i$.
3. Thus area of triangle $ACD = (1/2) * AD * CD = (1/2) * (64/17) * (120/17) = (1/2) * (64 * 120)/(17^2)$.
4. $555 < 625$, so highest power is $5^3 = 125$.
5. So $z1 = (a + 2i)/a = 1 + (2i)/a$.
6. We need to maximize $f(x) = 4(x + 7)(2 - x)$.
7. - above line x + y = 1 - above line y = x - 1 - below line y = x + 1 - within square.
8. Shoelace $sum_1 = x_i * y_{i+1} : -1 * 4 = -42 * (-4) = -82 * (-1) = -2 - 1 * 1 = -1Sum_1 = -4 - 8 - 2 - 1 = -15$
9. Let S = 3n+3 = 3(n+1).
10. Compute 5x+9 at x=-2: 5*(-2)+9 = -10+9 = -1.
11. Let's compute vectors.
12. For i=2, j=2: f(2,2)=f(1, f(2,1))=f(1,0)=2. Yes.
13. So remainder 3.
14. For x between 0 and 1, y between 1 and 0.
15. We need count of pairs (a,b) such that a*b divisible by 5.

GT: Non-removable | Activation Probe Correct: ×

1. not perfect square.
2. So quotient $x^2 - 4x - 12$.
3. Check if within interval [-1,1/2]. Yes.
4. Compute $a = sqrt(20^2 + 400) = sqrt(400 + 400) = sqrt(800) = 20 * sqrt(2) = 28.2843$.
5. That gives u = 39, 89, 139, etc.
6. - For $i \geq 1 : -ifj = 0 : f(i, 0) = f(i - 1, 1)$.
7. $G - M = ((b + c_x)/3 - 2b/3, c_y/3 - 0) = ((b + c_x - 2b)/3, c_y/3) = ((c_x - b)/3, c_y/3)$.
8. Wait $8^2 + 15^2 = 64 + 225 = 289 = 17^2$. Yes!
9. Equation same: $kx = (x + 2)^2$.
10. So $4x = 16y - 25$.
11. The problem says: The ratio of Denali's pay to Nate's pay would be the same if Denali started walking 4x more dogs and Nate stayed at 12 dogs or if x of Nate's dogs were reassigned to Denali.
12. Now Archimedes wants to minimize f(x).
13. sqrt(3481) =?
14. Next $8^{2/3}$.
15. Since 1985 is odd, we can compute: R = 1 - 2 + 3 - 4 + ... + 1985.

E.4 Reasoning step importance prediction results

In Section 4, we define and evaluate several methods for predicting reasoning step importance from tokens and latent representations. Due to budget constraints on the number of steps that could be processed through the external LLM-as-a-judge, we analyze this baseline on a limited evaluation set of randomly-chosen $1,500$ steps per model and dataset. The results of non-LLM-as-a-judge methods, presented in Figure 2, report evaluation accuracy on all available data. For an identical comparison,

we show in Figure 7 the results of the other methods when evaluated only on the $1,500$ steps. The results are consistent with those in Section 4 across all models and datasets.

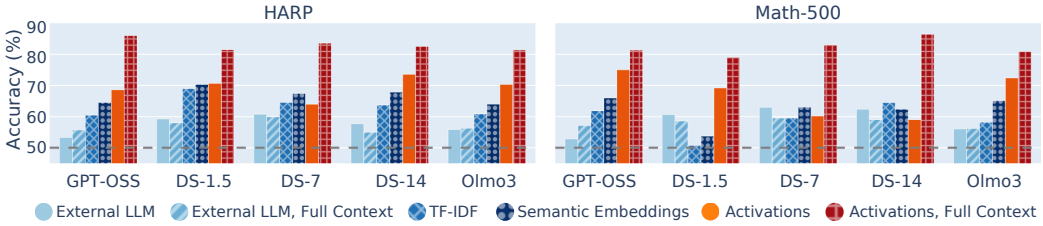


Figure 7: **Evaluation of all importance detection methods on the 1500-sized LLM-as-a-judge evaluation subset.**

E.5 Math-500 cross-model results

We show results for the cross-model analysis on the MATH-500 dataset in Figure 8. The results for MATH-500 are consistent with the results presented in Figure 3, supporting the claim that the notion of importance is universal across models.

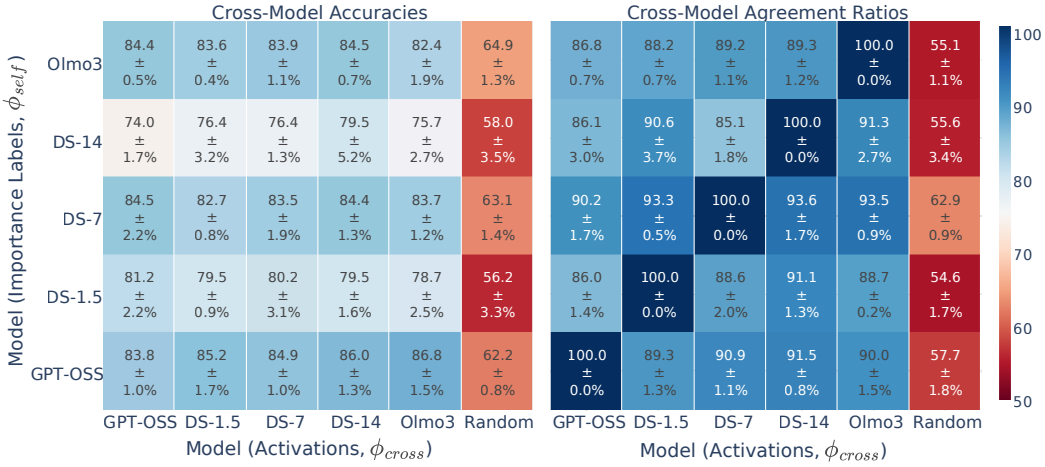


Figure 8: **Cross-model activation-based probe results on MATH-500.** Probes trained on one model's activations (ϕ_{cross}) are able to predict labels derived from another model (ϕ_{self}), and have high agreement ratios on individual labels with the probe trained when $\phi_{\text{cross}} = \phi_{\text{self}}$, suggesting the notion of importance is encoded in a universal manner across mathematical reasoning datasets.

E.6 Probe analysis results

Figures 9 to 12 present the probe analysis results on all models, as discussed in Section 5.2. Each figure contains three sub-plots. The left sub-plot shows accuracies of alternative probes—trained on activations from different token positions within each reasoning step, at different layers. The central sub-plot reports the correlation of probe predictions with surface-level features, which is consistently low across all models and probes. The right sub-plot compares probe accuracy against a regression model trained on all surface-level features: activation-based probes outperform this baseline by a large margin, further supporting the conclusion that the probes capture meaningful, non-surface-level structure.

E.7 Negative results

Several of our experiments yielded negative results that did not contribute to our understanding of the concept of reasoning step importance, yet we view them as worthwhile to mention.

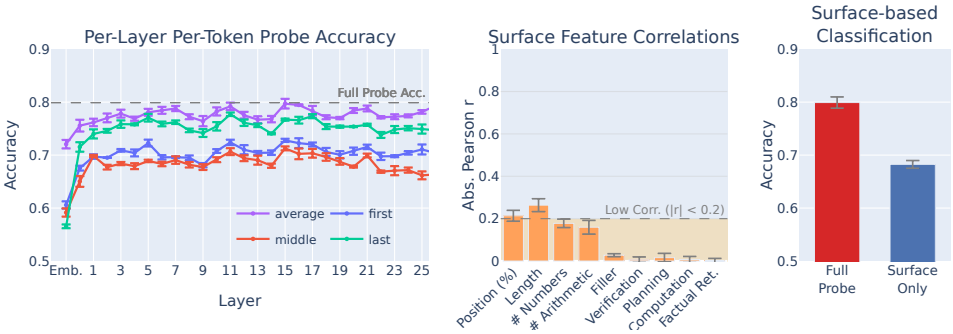


Figure 9: **Analysis of the importance probe in DeepSeek-1.5B (HARP dataset).**

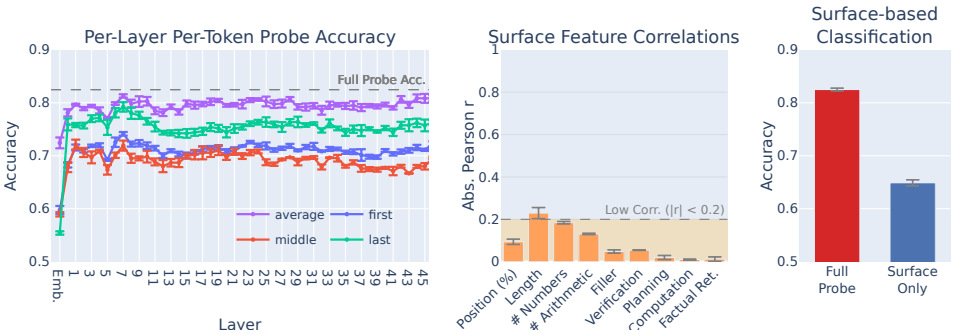


Figure 10: **Analysis of the importance probe in DeepSeek-14B (HARP dataset).**

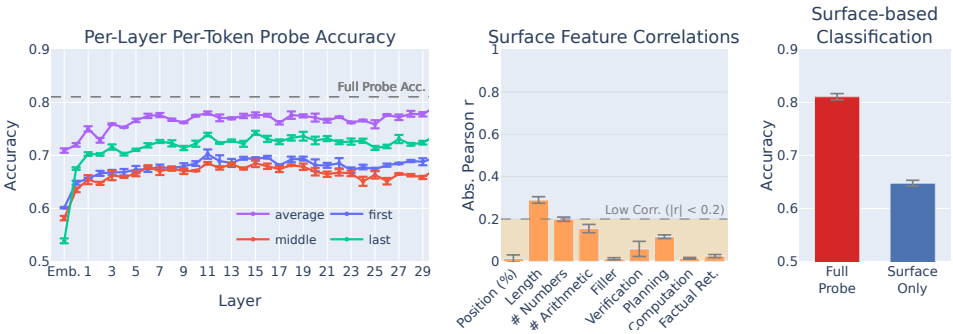


Figure 11: **Analysis of the importance probe in Olmo3-7B-Think (HARP dataset).**

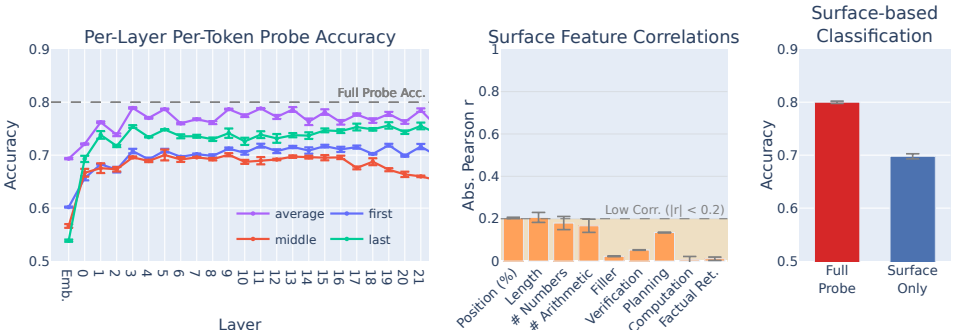


Figure 12: **Analysis of the importance probe in GPT-OSS-20B (HARP dataset).**

E.7.1 Different intervention types

To investigate whether specific reasoning steps have different functional roles in the reasoning process—and whether these relate to their importance—we design a set of counterfactual interventions applied to individual steps. Unlike the sufficiency evaluations in the main paper, where all steps except a candidate subsequence are removed, here we intervene on a *single* step at a time while leaving the rest of the chain intact.

For each target step, we apply four interventions:

1. **Removal**: the step is deleted entirely from the reasoning chain. This is the step-level parallel to the removal intervention from the main paper.
2. **Filler**: the step is replaced with repetitions of the filler sentence "Let me think..." padded with ". . ." tokens to match the original token count.
3. **Semantic paraphrase**: the step is replaced with a semantically equivalent restatement generated by an external LLM (Gemini-2.5-Flash), preserving meaning while varying its tokens.
4. **Context-averaged activation**: the step's activations are replaced with its context-averaged activations. To calculate the context-averaged activations, we append the target reasoning step to 10 randomly sampled reasoning chains, after a randomly chosen reasoning step. We calculate the target reasoning step's activations and average them across those contexts.

Across all model–dataset settings, the four interventions produce highly correlated outcomes. In a vast majority of cases, no single-step intervention has a meaningful effect on sufficiency. We attribute this to the inherent redundancy of reasoning chains: even if an important step is intervened on, the model can still recover the correct answer from other steps. These results further motivate our choice to intervene on reasoning chains in the reasoning subsequence resolution.

To measure a more fine-grained effect of these interventions, we measure the l_2 distance between the pre- and post-intervention activation tensors of reasoning steps subsequent to the intervened step. All four interventions again show consistent correlations with one another: when one intervention produces a large downstream perturbation, so do the others, this difference does not localize to specific step types in a way that supports a meaningful taxonomy. Removal and filler interventions generally produce larger perturbations than the paraphrase or context-averaged interventions, but these differences do not localize to specific steps in a way that supports a meaningful taxonomy of step types based on functionality.

E.7.2 Analysis of importance probes

To assess whether the probe weights encoded interpretable directions in the residual stream, we applied the logit lens (nostalgebraist, 2020) to the per-layer weight vectors of the layerwise-linear probe (each of shape $\mathbb{R}^d$). Across all model–dataset pairs and layers, the projected tokens were not semantically meaningful, suggesting the probes do not operate through easily human-interpretable vocabulary directions (See results for a subset of layers in Table 8).

Table 8: Logit lens analysis of specific layers in layerwise-linear probe weights. Each layer's weight vector is projected through the model's unembedding to identify the most positively (Top) and negatively (Bottom) associated tokens.

Layer	Direction	Tokens
0	Top	aea, agua, .cool, cko, cheon, elman, (lang, .random, .scroll, .arm
	Bottom	arre, mout, Jules, spl, ozn, _PUSH, blaze, Published, Written, Ment
3	Top	qat, eness, /modal, inam, nae, Quin, quin, yna, quiv, enk
	Bottom	arre, -Star, soak, URE, softer, tramp, Fle, cem, LEG, HG
7	Top	.fm, akus, idad, kau, -Bo, ista, .Serv, aua, Pier, alur
	Bottom	dise, goo, Invalidate, uret, Lap, Hond, ADVISED, vii, reneg, invalidate
11	Top	accommodating, _logic, circul, _UPLOAD, _plugins, imaginative, mou, mos, akter, konst
	Bottom	ysa, asal, DF, SF, Nether, incer, ELE, chl, APS, IPP
15	Top	PRES, -Str, -now, relate, pesc, UDIO, _GENERIC, TEC, prescribe, ICES
	Bottom	hein, vien, afr, etto, anoa, oron, %+, aign, court, Him
19	Top	MUT, alja, Haarlem, Hughes, Cartoon, heads, HQ, perfectly, accr, GBR
	Bottom	markdown, tala, mete, lade, .pattern, avana, pomp, ond, meteor, maq
23	Top	037, 770, 944, 071, tay, cind, 210, 799, taan, pone
	Bottom	Moc, [d, dj, (d, =d, d, Moj, hv, roller, Mog

F Reasoning chain examples

We present several examples for generated reasoning chains. For each reasoning chain, we mark shortcut steps that were removed pre-analysis in gray, removable reasoning steps in orange and non-removable reasoning steps in green. The minimal sufficient prefix, after which the model is able to generate the answer with high probability, is marked in **bold**. For presentation purposes, the examples shown here are relatively short reasoning chains (less than 20 reasoning steps and 1000 tokens). The full results, including the generated reasoning chains and the importance analysis results will be made publicly available.

Example 1 (GPT-OSS-20B, HARP)

Input: Solve the following problem. You MUST put your final answer in \boxed{}.
Problem: The smallest product one could obtain by multiplying two numbers in the set $\{-7, -5, -1, 1, 3\}$ is:

Reasoning Chain: We need to find smallest product. most negative? or smallest numeric value? Usually "smallest product" meaning minimal value, i.e., most negative. We have set of numbers: -7, -5, -1, 1, 3. Multiply any two numbers (could be same? Usually two numbers from set, maybe distinct? Not specified. Usually "multiplying two numbers in the set" could be any two numbers, maybe same allowed? But product of same number with itself? Let's consider both possibilities. We need smallest product. Let's compute all pairwise products. List numbers: -7, -5, -1, 1, 3. Compute products: -7 * -5 = 35, -7 * -1 = 7, -7 * 1 = -7, -7 * 3 = -21, -5 * -1 = 5, -5 * 1 = -5, -5 * 3 = -15, -1 * 1 = -1, -1 * 3 = -3, 1 * 3 = 3. Also consider squares: -7 * -7 = 49, -5 * -5 = 25, -1 * -1 = 1, 1*1=1, 3*3=9. Now find smallest value among all these: we have negative values: -21, -15, -7, -5, -3, -1. The smallest is -21. So product -21 from -7 * 3. Also -7 * 3 = -21. Also -5 * 3 = -15. So minimal is -21. But check if any product less than -21? No. So answer -21. But if we consider distinct numbers only, -7 * 3 is distinct, so still -21. So answer -21. Thus boxed answer: \boxed{-21}.

Example 2 (GPT-OSS-20B, HARP)

Input: Solve the following problem. You MUST put your final answer in \boxed{}.
Problem: For $p = 1, 2, \cdots, 10$ let S_p be the sum of the first 40 terms of the arithmetic progression whose first term is p and whose common difference is $2p - 1$; then $S_1 + S_2 + \cdots + S_{10}$ is:

Reasoning Chain: We need to compute S_p for each p from 1 to 10. Arithmetic progression: first term $a_1 = p$, common difference $d = 2p - 1$. Number of terms $n = 40$. Sum formula: $S_p = n/2 * (2a1 + (n - 1)d) = 40/2 * (2p + (39)(2p - 1))$. Let's compute: $20 * (2p + 39(2p - 1))$. Simplify inside: $2p + 39(2p - 1) = 2p + 78p - 39 = 80p - 39$. So $S_p = 20 * (80p - 39) = 1600p - 780$. Wait $20*80p = 1600p; 20*(-39) = -780$. So $S_p = 1600p - 780$. Now sum over $p = 1$ to 10: Sum $S = \sum_{\{} p = 1\}^{\{}10\}(1600p - 780) = 1600 \sum p - 780 * 10$. $\sum pfrom1to10 = 55$. So $1600 * 55 = 1600 * 50 + 1600 * 5 = 80,000 + 8,000 = 88,000$. Then minus $7800 = 80,200$? Wait $780 * 10 = 7,800$. So $88,000 - 7,800 = 80,200$. So answer 80200. Let's double-check arithmetic: $1600 * 55 = 1600 * (5 * 11)$?? Actually $55 = 5 * 11$. $1600 * 5 = 8000$; $8000 * 11 = 88,000$. Yes. Subtract $7,800$ gives $80,200$.

So final answer: 80200. Let's ensure no misinterpretation: S_p is sum of first 40 terms of AP with first term p and common difference $2p - 1$. Yes. So $S_p = 20 * (2p + 39(2p - 1))$. Let's recompute inside: $2p+39(2p-1) = 2p+78p-39 = 80p-39$. Good. Multiply by 20: $1600p-780$. Good. Sum $p = 1$ to 10: $1600*55-780*10 = 88,000-7,800 = 80,200$. Good. So answer: 80200. Provide in boxed form.

Example 3 (DeepSeek-R1-7B, HARP)

Input: Solve the following problem. You MUST put your final answer in \boxed{}.
Problem: Chandler wants to buy a 500 dollar mountain bike. For his birthday, his grandparents send him 50 dollars, his aunt sends him 35 dollars and his cousin gives him 15 dollars. He earns 16 dollars per week for his paper route. He will use all of his birthday money and all of the money he earns from his paper route. In how many weeks will he be able to buy the mountain bike?

Reasoning Chain: Okay, so Chandler wants to buy a mountain bike that costs $500. He gets some money from his grandparents, aunt, and cousin for his birthday, and he also earns some money from his paper route every week. The question is asking how many weeks he needs to work to earn enough money to buy the bike, using all his birthday money and all his weekly earnings. First, let me figure out how much birthday money Chandler has in total. His grandparents sent him $50, his aunt sent $35, and his cousin gave $15. So, I need to add those amounts together. Let me write that down: 50 + 35 + 15. Hmm, 50 plus 35 is 85, and then 85 plus 15 is 100. So, Chandler has $100 from his birthday gifts. Now, the bike costs $500, and he already has $100. So, I need to find out how much more money he needs. That would be 500 minus 100, which is 400. So, Chandler still needs $400. He earns $16 per week from his paper route. He's going to use all of his birthday money and all of his weekly earnings to buy the bike. So, the $400 he needs will come entirely from his weekly earnings because he's already used up all his birthday money. Now, to find out how many weeks he needs to work to earn $400 at $16 per week, I can set up an equation. Let me denote the number of weeks as 'w'. So, 16 times w equals 400. In equation form, that's 16w = 400. To solve for 'w', I need to divide both sides of the equation by 16. So, w equals 400 divided by 16. Let me calculate that. 400 divided by 16. Hmm, 16 times 25 is 400 because 16 times 20 is 320, and 16 times 5 is 80, so 320 plus 80 is 400. So, 16 times 25 is 400, which means w is 25. Wait, let me double-check that division to make sure I didn't make a mistake. 16 goes into 400 how many times? 16 times 10 is 160, so 400 minus 160 is 240. 16 times 15 is 240, so 10 plus 15 is 25. Yep, that's correct. So, 16 times 25 is indeed 400. Therefore, Chandler needs to work for 25 weeks to earn the remaining $400 after using his birthday money. So, in 25 weeks, he'll have enough to buy the mountain bike. Just to recap, Chandler has $100 from birthday gifts, needs $500 total, so he needs an additional $400. He earns $16 each week, so dividing 400 by 16 gives 25 weeks. That seems right. I don't think I made any mistakes here. The calculations are straightforward, and the division checks out. So, I'm confident that the answer is 25 weeks. **Final Answer** Chandler will be able to buy the mountain bike in \boxed{25} weeks.

Example 4 (Olmo-3-7B-Think, HARP)

Input: Solve the following problem. You MUST put your final answer in \boxed{}.
Problem: Chandra pays an on-line service provider a fixed monthly fee plus an hourly charge for connect time. Her December bill was $12.48, but in January her bill was $17.54 because she used twice as much connect time as in December. What is the fixed monthly fee?

Reasoning Chain: Okay, so I need to figure out the fixed monthly fee that Chandra pays. The problem says she has a fixed monthly fee plus an hourly charge for connect time. In December, her bill was $12.48, and in January, it was $17.54 because she used twice as much connect time as in December. Hmm, let me break this down. First, let me assign some variables to the unknowns. Let's say the fixed monthly fee is F dollars. That's the part that's the same every month, right? Then there's the hourly charge. Let me call the hourly rate H dollars per hour. So, the total bill each month is the fixed fee plus the hourly rate multiplied by the number of hours she used that month. In December, her bill was $12.48. Let me denote the number of hours she used in December as C hours. Then, the equation for December's bill would be: $F + H * C = 12.48$ Then in January, she used twice as much connect time, so that would be 2C hours. And her January bill was $17.54. So the equation for January would be: $F+H*(2C) = 17.54$ So now I have two equations: 1) $F+H*C = 12.48$ 2) $F + 2H * C = 17.54$ I need to solve these two equations to find F, the fixed monthly fee. Since we have two equations and two variables (F and $H * C$), maybe I can subtract the first equation from the second to eliminate F. Let me try that. Subtract equation 1 from equation 2: $(F + 2H * C) - (F + H * C) = 17.54 - 12.48$ Simplify the left side: $F - F + 2H * C - H * C = 17.54 - 12.48$ Which becomes: $H * C = 5.06$ So, H*C is $5.06. Wait, but H is the hourly rate and C is the number of hours in December. So $H * C$ is the variable cost for December's connect time. Then, the fixed fee F can be found by plugging back into equation 1. From equation 1: F + (H*C) = 12.48 We just found that H*C = 5.06, so substituting that in: $F + 5.06 = 12.48$ Therefore, $F = 12.48 - 5.06$ Let me compute that. 12.48 minus 5.06. Hmm, $12.48 - 5$ is 7.48, then subtract another 0.06, so 7.42. So F = $7.42? Wait, let me check the subtraction again to be sure. 12.48 minus 5.06: 12.48 -5.06 ---- Let's do it step by step. 12.48 minus 5 is 7.48. Then subtract the remaining 0.06: 7.48 - 0.06 = 7.42. Yeah, that's right. So F is $7.42? But let me verify with the second equation to make sure. If F is 7.42, then in January, the bill is F + 2*(H*C). We know H*C is 5.06, so 2*(H*C) is 10.12. Then 7.42 + 10.12 should equal 17.54. Calculating 7.42 + 10.12: 7 + 10 is 17, 0.42 + 0.12 is 0.54, so total is 17.54. Perfect, that matches the January bill. So that seems to check out. Therefore, the fixed monthly fee is $7.42. So I should put that in a box. Wait, but let me just make sure I didn't make any wrong assumptions. The problem says "a fixed monthly fee plus an hourly charge for connect time." So the total bill is fixed fee plus (hourly rate times hours used). That's how I set it up. So December: F + H*C = 12.48, January: F + H*(2C) =17.54. Then subtracting gives H*C = 5.06, so fixed fee is 12.48 -5.06=7.42. Then verification works. Seems solid. Is there another way to approach this? Maybe solve for one variable in terms of the other? From the first equation, F = 12.48 - H*C. Then plug that into the second equation: 12.48 - H*C + 2H*C = 17.54 Simplify: 12.48 + H*C =17.54 Then H*C =17.54 -12.48 =5.06, same as before. So then F=12.48 -5.06=7.42. Yep, same result.

So that's consistent. Therefore, I think that's the right answer. The fixed monthly fee is $7.42. So the answer is 7.42, in a box. **Final Answer** The fixed monthly fee is \boxed{7.42}.

Example 5 (DeepSeek-R1-7B, MATH-500)

Input: Solve the following problem. You MUST put your final answer in \boxed{}.
Problem: What is the integer value of x in the arithmetic sequence $3^2, x, 3^4$?

Reasoning Chain: Okay, so I have this problem here: I need to find the integer value of x in the arithmetic sequence $\mathbf{3^2, x, 3^4}$. Hmm, arithmetic sequence, right? Let me think about what that means. An arithmetic sequence is a sequence of numbers where the difference between consecutive terms is constant. So, if I have three terms in a row, the middle one is the average of the first and the third. That makes sense because in an arithmetic sequence, each term is obtained by adding a common difference. So, if I have the first term, then the second term is first term plus d, and the third term is second term plus d, which is first term plus 2d. So, in this case, the first term is $\mathbf{3^2}$, which is 9. The third term is $\mathbf{3^4}$, which is 81. So, the sequence is 9, x, 81. Since it's an arithmetic sequence, the difference between the first and second term should be the same as the difference between the second and third term. Let me write that down. The difference between the second term and the first term is $\mathbf{x - 9}$. The difference between the third term and the second term is $\mathbf{81 - x}$. Since these differences are equal, I can set up the equation: $\mathbf{x - 9 = 81 - x}$ Okay, so now I need to solve for x. Let me do that step by step. First, I can add x to both sides to get rid of the x on the right side. So, adding x to both sides: $\mathbf{x - 9 + x = 81 - x + x}$ Simplifying both sides: 2x - 9 = 81 Now, I can add 9 to both sides to isolate the term with x: $2x - 9 + 9 = 81 + 9$ Which simplifies to: $2x = 90$ Then, I can divide both sides by 2 to solve for x: $2x/2 = 90/2$ So, $x = 45$ Wait, let me check if that makes sense. If x is 45, then the sequence is $9, 45, 81$. Let's see if the differences are the same. $45 - 9 = 36$, $81 - 45 = 36$ Yes, both differences are 36, so that works. So, x is 45. But hold on, the problem says to put the final answer in a box and as an integer. 45 is an integer, so that should be fine. Just to make sure I didn't make any mistakes, let me go through the steps again. First term: $3^2 = 9$ Third term: $3^4 = 81$ In an arithmetic sequence, the middle term is the average of the first and third terms. So, x should be $(9 + 81)/2$. Let me calculate that: $(9 + 81) = 90$ 90 divided by 2 is 45. Yep, that's the same answer I got before. So, that confirms it. I think that's solid. I don't see any mistakes in my reasoning. So, x is definitely 45. **Final Answer** The integer value of x is \boxed{45}.