When LLMs Know They Don't: Probing Latent Representations for Logical Insufficiency

Matt Wang[*]
Center for Language and Speech Processing
Johns Hopkins University, Baltimore, MD
`hwang302@jh.edu`

Prabhav Singh[*]
Center for Language and Speech Processing
Johns Hopkins University, Baltimore, MD
`psingh54@jh.edu`

Tom Wang[*]
Center for Language and Speech Processing
Johns Hopkins University, Baltimore, MD
`xwang397@jh.edu`

Abstract

Large language models (LLMs) confidently hallucinate on logically insufficient questions—problems lacking necessary information for deterministic solutions—silently assuming missing constraints rather than requesting clarification. We investigate whether this failure reflects the LLMs inability to recognize insufficiency or merely their inability to verbalize this internal knowledge. Through systematic probing experiments across three mathematical reasoning benchmarks (UMWP, GSM8K Insufficient, TreeCut) and four LLMs (1.5B–7B parameters), we demonstrate that logical insufficiency is robustly encoded as a linearly separable property in frozen LLM representations. Simple linear probes achieve 80–91% F1 on binary insufficiency detection and generalize across in-domain datasets, despite models' verbal self-assessment accuracy remaining near chance. This *representation-language gap averaging +25.6 percentage points* reveals that models already "know" when they cannot know, but this knowledge remains latent. We exploit these internal signals to implement lightweight probe-guided intervention systems that detect insufficient queries before generation, offering a training-free pathway to epistemic humility without extensive fine-tuning or post-hoc verification. Further, we investigate the performance of these LLMs in **verbalizing insufficiencies and asking for the right information** once the *said insufficiency* is detected.

1 Introduction

Large language models (LLMs) have achieved strong performance on mathematical reasoning benchmarks [9, 12], yet exhibit a critical failure: *confident hallucination on logically insufficient questions*. When presented with problems lacking necessary information—such as "Alice has some apples and Bob has twice as many. How many do they have together?"—models rarely request clarification [28, 21]. Instead, they silently assume missing constraints and generate plausible but unfounded answers. We focus on *logical insufficiency*: cases where problem structure is well-defined

[*]Authors contributed equally to this work.

Preprint. Under review.

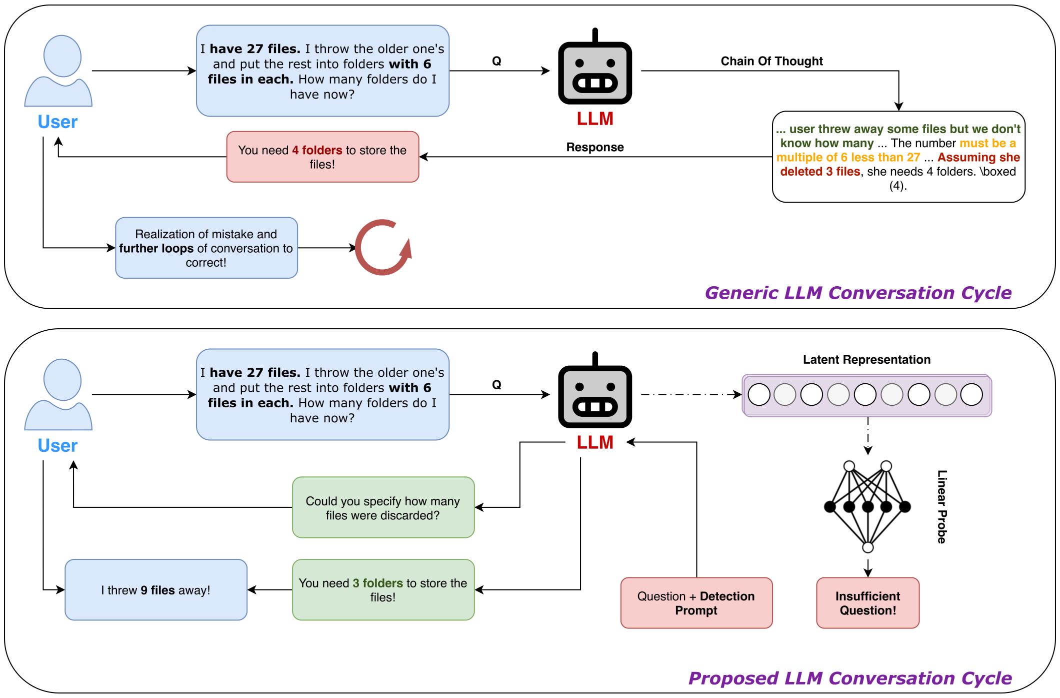

User
I have 27 files. I throw the older one's and put the rest into folders with 6 files in each. How many folders do I have now?
Q
LLM
Chain Of Thought
... user threw away some files but we don't know how many ... The number must be a multiple of 6 less than 27 ... Assuming she deleted 3 files, she needs 4 folders. \boxed (4).
Response
You need 4 folders to store the files!
Realization of mistake and further loops of conversation to correct!
Generic LLM Conversation Cycle
User
I have 27 files. I throw the older one's and put the rest into folders with 6 files in each. How many folders do I have now?
Q
LLM
Latent Representation
Linear Probe
Insufficient Question!
Question + Detection Prompt
Could you specify how many files were discarded?
You need 3 folders to store the files!
I threw 9 files away!
Proposed LLM Conversation Cycle

Figure 1: **Probe-Guided Clarification Pipeline.** *Top:* Standard LLM behavior confidently answers insufficient questions, hallucinating missing information. *Bottom:* Our approach extracts latent representations from the question, uses trained probes to detect insufficiency, and prompts the model to request exactly the missing information before generation, preventing hallucination at inference time.

but critical information is absent, making multiple answers equally valid.[2] This overconfidence poses deployment risks where users receive conditionally correct but misleading responses [13].

Current hallucination detection methods focus on *post-hoc verification*: sampling multiple outputs [18, 7], querying knowledge bases [22], or employing auxiliary judges [8, 20]. These incur substantial computational overhead and struggle with logical insufficiency, where generation cannot resolve missing input information [33]. Uncertainty quantification work [14, 34, 17, 4, 16] analyzes output-level signals (token probabilities, verbalized confidence) rather than investigating whether models internally represent sufficiency. Unanswerable question detection [24, 15, 25] typically requires supervised training and does not examine whether pre-trained LLMs already encode this information before generation.

Mechanistic interpretability research reveals that LLMs encode semantic properties like truthfulness [19, 3], sentiment [30] and, entity attributes [11] as *linearly separable structures* in hidden states. Linear probes achieve 80–90% accuracy on binary tasks [2, 5], suggesting a "representation-language gap" [10]: models maintain internal representations they do not verbalize. Activation steering [32, 36] demonstrates that intervening on representations can alter behavior without modifying weights. For hallucination, methods leveraging internal states [27, 26] focus on *post-generation* factual inconsistencies rather than *pre-generation* insufficiency detection.

Do LLMs internally recognize when queries are logically insufficient, even when failing to express this uncertainty? If models encode insufficiency in hidden states, overconfident hallucination reflects a *verbalization* failure, not recognition failure. This opens the door to lightweight interventions without fine-tuning and gives rise to three questions we attempt to answer:

[2]We distinguish logical insufficiency from general ambiguity. "What's the capital?" is ambiguous due to under-specified context. "A triangle has sides 10cm and 15cm. What is its area?" is logically insufficient. The structure is clear, but a necessary constraint (included angle or third side) is missing for a unique solution.

RQ1: Linear Separability and Mechanistic Structure. Following recent work demonstrating linear structure in truth representations [19, 3], we ask whether binary sufficiency/insufficiency is encoded as a linearly separable property in frozen LLM representations. We train logistic regression probes on hidden states from all transformer layers to identify (1) whether insufficiency can be decoded with high accuracy, (2) at which layers this signal emerges most strongly, and (3) what geometric properties these representations exhibit. Prior work on semantic property extraction [30, 11] suggests abstract concepts crystallize in late layers, but insufficiency may exhibit different localization patterns.

RQ2: Representation-Language Gap and Cross-Dataset Generalization. We directly compare probe-based detection against models' zero-shot verbal self-assessment by prompting them to judge question answerability.[3] This quantifies the *representation-language gap* [10, 37]: the performance difference between what models internally represent and what they explicitly verbalize. A large gap would confirm that insufficiency knowledge exists but remains latent. We further test cross-dataset transfer [5] by training probes on each benchmark individually and on combined data, then evaluating on held-out datasets within the mathematical reasoning domain. Strong generalization would indicate domain-general sufficiency signals rather than dataset-specific artifacts [28, 21].

RQ3: Fine-Grained Knowledge of Missing Information. Beyond detecting *whether* information is missing, we investigate whether models internally represent *what* information is missing. Logical insufficiency manifests in multiple forms: missing critical values, contradictory constraints, inappropriate question types, and others [28]. We extend our binary probes to 6-class classification on UMWP's structured insufficiency taxonomy (5 insufficiency types plus sufficient class), then evaluate whether probe-guided prompting can elicit targeted clarification questions that correctly identify the specific missing constraint. This distinction is crucial for practical deployment: generic uncertainty expressions provide limited utility, while precise identification of missing information enables productive human-AI collaboration.

Our work makes three contributions. **First**, we conduct comprehensive probing analysis demonstrating that logical insufficiency is robustly encoded as linearly separable features in frozen LLM representations, with layer-wise and geometric analysis revealing where and how this signal emerges. **Second**, we quantify the representation-language gap through controlled comparison of probe-based detection versus verbal self-assessment, and demonstrate strong cross-dataset generalization alongside successful extension to fine-grained multi-class insufficiency type classification. **Third**, we implement and evaluate a training-free probe-guided intervention pipeline (Figure 1) that detects insufficient queries at inference time and prompts models to request exactly the missing information before generation, providing a practical pathway from interpretability insights to reliable deployment without post-hoc verification or auxiliary models.

The paper is organized as follows: Section 2 describes our three mathematical reasoning benchmarks and the systematic procedures for generating insufficient variants. Section 3 details our experimental methodology, including model selection, probe architectures, training procedures, and evaluation protocols. Section 4 presents our main experimental results, organized into four subsections: linear probe feasibility and mechanistic analysis (§4.1), cross-dataset generalization and verbalization gap (§4.2), fine-grained multi-class classification (§4.3), and training-free intervention (§4.4). Section 5 discusses implications and limitations. Section 6 concludes.

2 Datasets and Models

We evaluate our approach on three mathematical reasoning benchmarks specifically designed to test insufficiency detection capabilities. Each benchmark provides paired sufficient and insufficient variants, enabling controlled comparison of model representations. Table 1 summarizes the composition and splits across all three datasets.

[3] Verbal assessment uses a two-stage pipeline: we first prompt the LLM to assess whether the problem can be solved with the given information, then employ GPT-4o-mini as a judge to classify the LLM's response into binary answerability labels. See Appendix C for complete prompts and methodology.

Table 1: Dataset composition and splits. All datasets maintain balanced sufficient/insufficient ratios in test sets. UMWP provides fine-grained 6-class labels (5 insufficiency types + sufficient) for multi-class classification experiments.

Dataset	Train	Test	Total	Construction	Fine-Grained
UMWP	4,160	1,040	5,200	Human Expert	6 classes
TreeCut	14,930	1,040	15,970	Synthetic	Binary
GSM8K-Insuff.	1,632	408	2,040	GPT-4o Transform	Binary

2.1 UMWP: Unanswerable Math Word Problems

The UMWP (Unanswerable Math Word Problems) benchmark [28] provides 5,200 grade-school math problems with balanced sufficient/insufficient splits.[4] Each insufficient problem is manually constructed by domain experts who remove critical numerical values or constraints, creating five insufficiency categories: (1) missing critical information (e.g., omitting total count when asking for percentages), (2) incomplete constraint information (e.g., missing consumption rate in resource planning), (3) contradictory conditions (e.g., logically inconsistent constraints), (4) inappropriate question types (e.g., requesting information not derivable from context), and (5) missing intermediate results (e.g., required sub-calculations unavailable). We use the standard 80/20 train-test split (4,160 train, 1,040 test), maintaining class balance in both partitions.

2.2 TreeCut: Synthetic Unanswerable Problems

TreeCut [21] provides a scalable synthetic benchmark where problems are represented as dependency trees with variables as nodes and mathematical relationships as edges.[5] Insufficient variants are generated by systematically removing edges along the solution path, creating controlled insufficiency with known ground-truth missing constraints. Unlike UMWP's multi-class taxonomy, TreeCut provides binary labels (answerable/unanswerable), but its structured generation process ensures each insufficient example has an explicit missing edge that can be identified. Recent evaluations show TreeCut induces hallucinations in frontier models (GPT-4o: 64%, o3-mini: 44% worst-case), making it a challenging testbed for both detection and intervention. We sample 14,930 training examples and 1,040 test examples from the full synthetic distribution, matching UMWP test set size for fair comparison.

2.3 GSM8K-Insufficient: Programmatically Generated Variants

We create GSM8K-Insufficient by applying controlled transformations to a subset of the GSM8K benchmark [9], which originally contains 8,500 grade-school math problems with step-by-step solutions. To manage computational costs while ensuring high-quality transformations, we sample approximately 2,000 problems and use GPT-4o as a transformation engine to programmatically generate insufficient variants by removing exactly one critical numerical value from each problem (Appendix A details the complete prompt). We process both training and test splits with 50% insufficiency ratio, resulting in 1,632 training examples and 408 test examples after filtering for successful GPT-4o transformations and combining with sufficient variants.[6] This dataset tests generalization to a different construction methodology—programmatic GPT-4o transformation versus human expert annotation (UMWP) or structural synthesis (TreeCut).

2.4 Model Selection and Baseline Performance

We evaluate four pre-trained LLMs spanning 1.5B to 7B parameters: **Qwen2.5-Math-1.5B** [35], a mathematics-specialized model with 28 transformer layers; **Qwen2.5-1.5B-Instruct** [29], a general-purpose instruction-tuned variant with 28 layers; **Llama-3.2-3B-Instruct** [1], Meta's multilingual dialogue model with 16 layers; and **Qwen2.5-Math-7B** [35], the largest math-specialized variant

[4]Available at `https://github.com/Yuki-Asuuna/UMWP`

[5]Available at `https://huggingface.co/datasets/jouyang/treecut-math`

[6]We applied 80/20 train-test split after transformation to maintain consistency with other benchmarks and ensure proper held-out evaluation.

Table 2: Model accuracy on sufficient questions from each dataset's test set, sorted by average performance (↓). Models demonstrate strong solving capability when information is complete, with math-specialized variants outperforming general-purpose models. TreeCut's low accuracy reflects its structural complexity and departure from standard problem phrasing. Evaluation uses GPT-4o-mini as automated judge with strict numerical matching.

Model	**UMWP**	**GSM8K**	**TreeCut**	**Avg. ↓**
Qwen2.5-Math-7B	95.0%	91.2%	31.9%	**72.7%**
Qwen2.5-Math-1.5B	92.3%	87.0%	16.5%	65.3%
Llama-3.2-3B-Instruct	83.1%	78.2%	18.8%	60.0%
Qwen2.5-1.5B-Instruct	72.7%	68.9%	10.0%	50.5%

with 28 layers and 3,584 hidden dimensions. All models use standard decoder-only architectures and are evaluated in their publicly available pre-trained checkpoints without fine-tuning.

To establish baseline solving capabilities, we first evaluate models on *sufficient* questions from each dataset's test set.[7] Table 2 reports accuracy on sufficient questions, demonstrating that models exhibit strong mathematical reasoning when adequate information is provided. Math-specialized models (Qwen2.5-Math series) substantially outperform general-purpose models, with the 7B variant achieving 95.0% accuracy on UMWP and 91.2% on GSM8K. Notably, all models struggle on TreeCut sufficient questions (10.0–31.9% accuracy), reflecting the benchmark's structural complexity and departure from typical grade-school problem phrasing. This performance profile confirms models possess sufficient mathematical competence to solve problems when information is complete, isolating insufficiency detection as the core challenge rather than general reasoning ability.

3 Method

Our methodology consists of three components: (1) extracting frozen representations from pre-trained LLMs, (2) training lightweight linear probes to classify insufficiency from these representations, and (3) evaluating cross-dataset generalization and verbalization gaps. We describe each component below.

3.1 Representation Extraction

For each question q in our datasets, we extract hidden state representations from all layers of a frozen LLM without updating model parameters. Let $\mathbf{h}_\ell \in \mathbb{R}^{T \times d}$ denote the hidden states at layer $\ell \in \{0, 1, \dots, L\}$, where T is the sequence length and d is the hidden dimension.[8] We apply pooling to obtain a fixed-dimensional representation $\mathbf{z}_\ell \in \mathbb{R}^d$ for each layer.

Last Token Pooling. Our primary approach uses **last token pooling**, extracting the representation corresponding to the final token in the sequence:

$$\mathbf{z}_\ell = \mathbf{h}_\ell[T, :] \in \mathbb{R}^d \tag{1}$$

This choice follows standard practice in decoder-only language models [23, 6], where the last token aggregates information from the entire context via causal attention. For autoregressive models, the last token position has attended to all prior tokens and serves as the natural summary representation for sequence-level classification tasks.

[7]We use GPT-4o-mini as an automated judge to compare model-generated answers against ground-truth solutions, employing strict numerical matching with tolerance $\epsilon = 10^{-2}$ for floating-point values. See Appendix B for complete judging protocol.

[8]Layer 0 corresponds to the embedding layer. For our models, $L = 28$ (Qwen2.5-Math-1.5B, Qwen2.5-1.5B-Instruct, Qwen2.5-Math-7B) and $L = 16$ (Llama-3.2-3B-Instruct), with hidden dimensions $d \in \{1536, 2048, 3584\}$ depending on the model.

Mean Pooling. We additionally experiment with **mean pooling**, averaging hidden states across all token positions:

$$\mathbf{z}_\ell = \frac{1}{T}\sum_{t=1}^{T} \mathbf{h}_\ell[t,:] \in \mathbb{R}^d \tag{2}$$

Mean pooling provides an alternative aggregation that weights all positions equally, potentially capturing distributed information across the sequence. While we report mean pooling results for completeness, our analysis (§4.1) demonstrates that last token pooling consistently achieves superior or comparable performance across all models and datasets, confirming it as the more effective strategy for this task.

3.2 Linear Probe Architecture and Training

Following established methodology in mechanistic interpretability [2, 5, 19], we train **linear probes**—simple classifiers applied to frozen representations—to test whether insufficiency is linearly decodable. For binary classification, we use logistic regression, which learns a linear decision boundary in representation space.

Formally, given a representation $\mathbf{z}_\ell \in \mathbb{R}^d$ at layer ℓ and a binary label $y \in \{0, 1\}$ (0 = sufficient, 1 = insufficient), the probe predicts:

$$P(y = 1 \mid \mathbf{z}_\ell) = \sigma(\mathbf{w}_\ell^\top \mathbf{z}_\ell + b_\ell) \tag{3}$$

where $\mathbf{w}_\ell \in \mathbb{R}^d$ is the weight vector, $b_\ell \in \mathbb{R}$ is the bias term, and $\sigma(\cdot)$ is the sigmoid function. Parameters $\{\mathbf{w}_\ell, b_\ell\}$ are learned independently for each layer ℓ by minimizing the binary cross-entropy loss with L2 regularization:

$$\mathcal{L}(\mathbf{w}_\ell, b_\ell) = -\frac{1}{N}\sum_{i=1}^{N}\Big[y_i \log P(y_i = 1 \mid \mathbf{z}_\ell^{(i)}) + (1 - y_i)\log P(y_i = 0 \mid \mathbf{z}_\ell^{(i)})\Big] + \lambda\|\mathbf{w}_\ell\|_2^2 \tag{4}$$

where N is the number of training examples and $\lambda = 1.0$ is the regularization strength (inverse of scikit-learn's C parameter).

Training Procedure. We train probes using the LBFGS optimizer with maximum 3,000 iterations and convergence tolerance 10^{-4}. To address class imbalance in individual datasets, we employ balanced class weights: $w_{\text{class}} = \frac{N}{2 \cdot N_{\text{class}}}$, where N_{class} is the number of examples in each class.[9] All probes are trained on frozen representations extracted once and cached to disk, eliminating redundant forward passes.

Layer-wise Probing. Critically, we train **independent probes for every layer** $\ell \in \{0, 1, \ldots, L\}$ of each model. This layer-wise analysis reveals where in the network insufficiency representations emerge and how they evolve across depth [3, 30]. By comparing performance across layers, we identify which transformer layers encode the strongest insufficiency signals, with implications for both mechanistic understanding and computational efficiency of interventions.

For each configuration, we report standard classification metrics: accuracy, precision, recall, and macro F1 score (harmonic mean of precision and recall). F1 serves as our primary metric as it balances both false positives (hallucinating insufficiency) and false negatives (missing true insufficiency), both of which have deployment consequences.[10]

4 Experiments and Results

We present our findings in four parts: (§4.1) linear probe feasibility and mechanistic analysis demonstrating strong decodability of insufficiency, (§4.2) cross-dataset generalization and the representation-language gap, (§4.3) fine-grained multi-class insufficiency type classification, and (§4.4) training-free intervention results.

[9]This ensures the loss contribution from each class is weighted equally, preventing the probe from exploiting class priors.

[10]High precision ensures we do not unnecessarily interrupt users with clarification requests, while high recall ensures we catch genuinely insufficient queries before hallucination occurs.

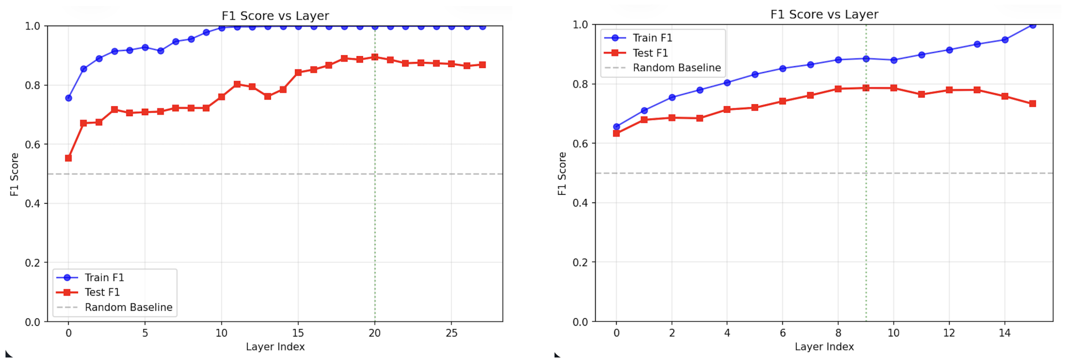
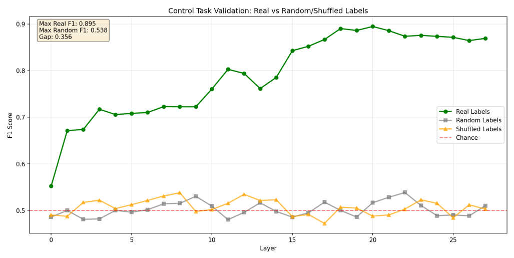
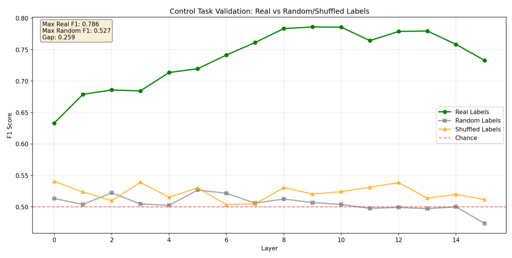


Figure 2: Layer-wise F1 scores for linear probes on UMWP. **Left**: Qwen2.5-Math-1.5B (28 layers) achieves peak F1 of 89.5% at layer 20. **Right**: Llama-3.2-1B-Instruct (16 layers) achieves peak F1 of 78.6% at layer 9. Both models show weak early-layer signals that strengthen and plateau in late layers, confirming that insufficiency is robustly encoded as a linearly separable property.



(a) Qwen2.5-Math-1.5B



(b) Llama-3.2-1B-Instruct

Figure 3: Control task validation. Probes trained on randomly permuted or shuffled labels achieve $\sim$50% F1 (chance), while real labels yield 89.5% (Qwen) and 78.6% (Llama). The large gap confirms that probes recover genuine insufficiency features, not spurious correlations.

4.1 Linear Probe Feasibility and Mechanistic Analysis

We first validate our central hypothesis: **logical insufficiency is linearly separable in frozen LLM representations**. For mechanistic interpretability analysis, we focus on two models—Qwen2.5-Math-1.5B (28 layers) and Llama-3.2-1B-Instruct (16 layers)—to conduct detailed layer-wise and geometric analyses. Subsequent subsections (§4.2–4.4) report results across all four models.

Layer-wise Performance and Linear Sufficiency. Figure 2 shows test F1 scores across all layers using linear probes (logistic regression) on UMWP. Both models exhibit a consistent trend: insufficiency signals are weak in early layers (F1 $\approx$ 55–65%), strengthen rapidly in middle layers, and plateau in late layers. For Qwen2.5-Math-1.5B, performance peaks at layer 20 with 89.5% F1. For Llama-3.2-1B-Instruct, the peak occurs earlier at layer 9 (78.6% F1), likely due to its shallower architecture (16 vs. 28 layers). These results far exceed our initial 70% F1 feasibility threshold and demonstrate that insufficiency awareness emerges gradually across network depth [3].

Control Task Validation. To rule out dataset artifacts (e.g., length biases, lexical cues), we conduct control experiments training probes on (1) true labels, (2) randomly permuted labels, and (3) shuffled labels. Figure 3 shows that on true labels, probes achieve 89.5% F1 (Qwen) and 78.6% F1 (Llama), but on random/shuffled labels, performance collapses to 50–54% F1 (chance level) across all layers. The large gap between real and random performance (35.6% for Qwen, 25.9% for Llama) demonstrates that probes recover genuine semantic features of insufficiency, not spurious statistical regularities [5].

Beyond measuring probe accuracy, we characterize the geometric and algebraic structure of insufficiency representations. We apply PCA and UMAP dimensionality reduction to visualize latent space geometry, and conduct subspace dimensionality analysis by training probes on PCA-reduced embeddings with varying numbers of components [19, 3] making it amenable to lightweight, interpretable intervention. Detailed visualizations are provided in Appendix D.

4.2 Cross-Dataset Generalization and Representation-Language Gap

Having established that insufficiency is linearly decodable within individual datasets (§4.1), we now investigate two critical questions: (1) do these representations generalize across different problem distributions, and (2) can models verbally express their internal knowledge of insufficiency?

Cross-Dataset Generalization. A central question is whether linear separability observed on one dataset represents a dataset-specific artifact or a more universal property of model representations. To evaluate robustness, we conduct a systematic cross-dataset transfer study: for each of the three datasets (UMWP, GSM8K-Insufficient, TreeCut), we train linear probes on one dataset's training split and evaluate on all three test sets, plus an additional configuration training on the pooled union of all datasets ("ALL").[11]

Table 3 reports cross-dataset F1 scores for all four models using last token pooling. Results reveal strong generalization across natural problem distributions: probes trained on UMWP or GSM8K achieve 80–92% F1 when tested on each other (off-diagonal cells), despite differences in problem phrasing, complexity, and construction methodology. This suggests that **the latent features encoding logical sufficiency are largely shared across realistic mathematical problem styles**, rather than being tied to dataset-specific surface patterns [5].

Generalization to TreeCut is noticeably weaker but still substantial (52–71% F1 for UMWP/GSM8K $\rightarrow$ TreeCut transfer). We hypothesize two contributing factors. First, TreeCut problems are synthetically constructed via tree-structured variable deletion [21], making them stylistically less natural than human-written UMWP or programmatically transformed GSM8K problems. Second, TreeCut introduces structural insufficiency mechanisms (missing dependency edges) that may differ geometrically from semantic omissions (missing numerical values) in natural problems. Nonetheless, cross-dataset performance far exceeds chance (50%), confirming that insufficiency signals transfer across heterogeneous domains.

The probe trained on all three datasets ("ALL" column, shaded) consistently matches or outperforms single-dataset probes across all test sets. This indicates that **a single linear probe can capture a multi-domain insufficiency direction**, even when the forms of missingness differ in structure [19]. The generalization pattern is consistent across model scales (1.5B to 7B) and architectures (Qwen vs. Llama), suggesting that linear sufficiency representations are an emergent and transferable property of pretrained LLMs rather than an artifact of probe fitting or dataset design. Mean pooling results (Appendix E) show qualitatively similar transfer patterns with slightly degraded absolute performance, confirming last token pooling as the superior aggregation strategy.

The Representation-Language Gap. Having demonstrated that models internally encode insufficiency as a robust, generalizable signal, we directly compare this latent knowledge against models' ability to verbally express uncertainty. We measure the **representation-language gap**: $\Delta = \text{Acc}_{\text{probe}} - \text{Acc}_{\text{verbal}}$, quantifying the performance difference between probing frozen representations versus prompting for explicit self-assessment.

For verbal baseline, we employ a two-stage zero-shot pipeline (Appendix C): we first prompt each model to assess whether the problem can be solved with the given information, requesting structured output via `\boxed{}` notation. We then use GPT-4o-mini as a judge to classify the model's response into binary answerability labels (YES = sufficient, NO = insufficient), ensuring robust evaluation even when models produce verbose or hedged responses.[12] For probe baseline, we report test accuracy from the best-performing layer when trained on each dataset (in-distribution upper bound).

Table 4 reveals a consistent and substantial gap across all 9 model-dataset pairs, averaging **+25.6 percentage points**. This yields two critical insights. First, **internal knowledge exceeds verbal**

[11]The ALL configuration combines 4,160 + 1,632 + 14,930 = 20,722 training examples from UMWP, GSM8K, and TreeCut respectively.

[12]This two-stage approach is necessary because models frequently generate extraneous explanations or non-standard formatting rather than clean binary answers. The GPT-4o-mini judge extracts the intended verdict from such responses. Complete prompts are provided in Appendix C.

Table 3: Cross-dataset generalization with last token pooling (F1 scores). Rows indicate training dataset, columns indicate test dataset. Diagonal cells (bold) show in-distribution performance; off-diagonal cells show transfer. ALL column (shaded) shows probes trained on combined data from all three datasets. Strong transfer between UMWP and GSM8K (80–94% F1) demonstrates that insufficiency representations generalize across natural problem distributions. TreeCut transfer is weaker but substantial (52–84% F1), reflecting its synthetic construction. Training on ALL datasets yields robust multi-domain probes.

Model	**Train \ Test**	**UMWP**	**GSM8K**	**TreeCut**	**ALL**
Qwen2.5-Math-1.5B	UMWP	**88.7**	87.2	61.6	87.1
	GSM8K	82.1	**90.0**	67.6	88.8
	TreeCut	57.6	52.0	**76.0**	74.2
	ALL	87.1	88.8	74.2	**83.4**
Qwen2.5-1.5B	UMWP	**88.9**	88.3	67.3	87.7
	GSM8K	79.5	**91.7**	66.7	90.5
	TreeCut	68.9	68.9	**80.5**	79.5
	ALL	87.7	90.5	79.5	**85.9**
Llama-3.2-3B	UMWP	**80.2**	82.9	66.6	81.2
	GSM8K	72.4	**85.8**	67.2	81.6
	TreeCut	65.1	61.8	**80.4**	78.0
	ALL	81.2	81.6	78.0	**80.3**
Qwen2.5-Math-7B	UMWP	**92.1**	88.5	65.2	86.4
	GSM8K	85.8	**94.0**	68.0	88.2
	TreeCut	70.6	69.0	**78.8**	83.1
	ALL	86.4	88.2	83.1	**85.9**

Table 4: The Representation-Language Gap. Comparison of zero-shot verbal accuracy (prompting for explicit self-assessment) vs. linear probe accuracy (accessing internal representations from best-performing layer). Δ indicates the accuracy gain achieved by probing latent knowledge directly. Gap averages +25.6 percentage points across all settings, demonstrating that models internally represent insufficiency far better than they can verbally express.

Model	**Dataset**	**Verbal Acc.**	**Probe Acc.**	**Gap (Δ)**	**Avg. Gap**
Qwen2.5-1.5B	UMWP	52.7%	88.8%	+36.1%	+36.3%
	GSM8K	49.5%	91.9%	+42.4%	
	TreeCut	50.0%	80.3%	+30.3%	
Qwen2.5-Math-1.5B	UMWP	61.2%	89.2%	+28.0%	+28.2%
	GSM8K	59.8%	90.4%	+30.6%	
	TreeCut	50.4%	76.2%	+25.8%	
Qwen2.5-Math-7B	UMWP	84.0%	91.7%	+7.7%	+12.3%
	GSM8K	88.7%	94.1%	+5.4%	
	TreeCut	57.0%	80.7%	+23.7%	
				Average Gap across all settings:	**+25.6%**

ability: In every setting, probing outperforms prompting. Notably, the base Qwen2.5-1.5B model performs near random chance verbally (∼50%) but achieves high latent accuracy (80–92%), suggesting instruction tuning is required to *articulate* insufficiency, not to *learn* it [14, 34]. Second, **scaling is not enough**: While the 7B model closes the gap on standard domains (UMWP: +7.7%, GSM8K: +5.4%), it retains a massive +23.7% gap on TreeCut. Even large models with strong verbal calibration on natural distributions fail to generalize their articulation abilities to synthetic edge cases, despite their internal representations transferring successfully (Table 3).

These findings confirm that **the failure to identify unsolvability is a failure of reporting, not recognition** [10, 37]. Models already "know" when they cannot know, but this knowledge remains latent and unexpressed—a gap that motivates intervention strategies (§4.4) to surface internal signals before generation.

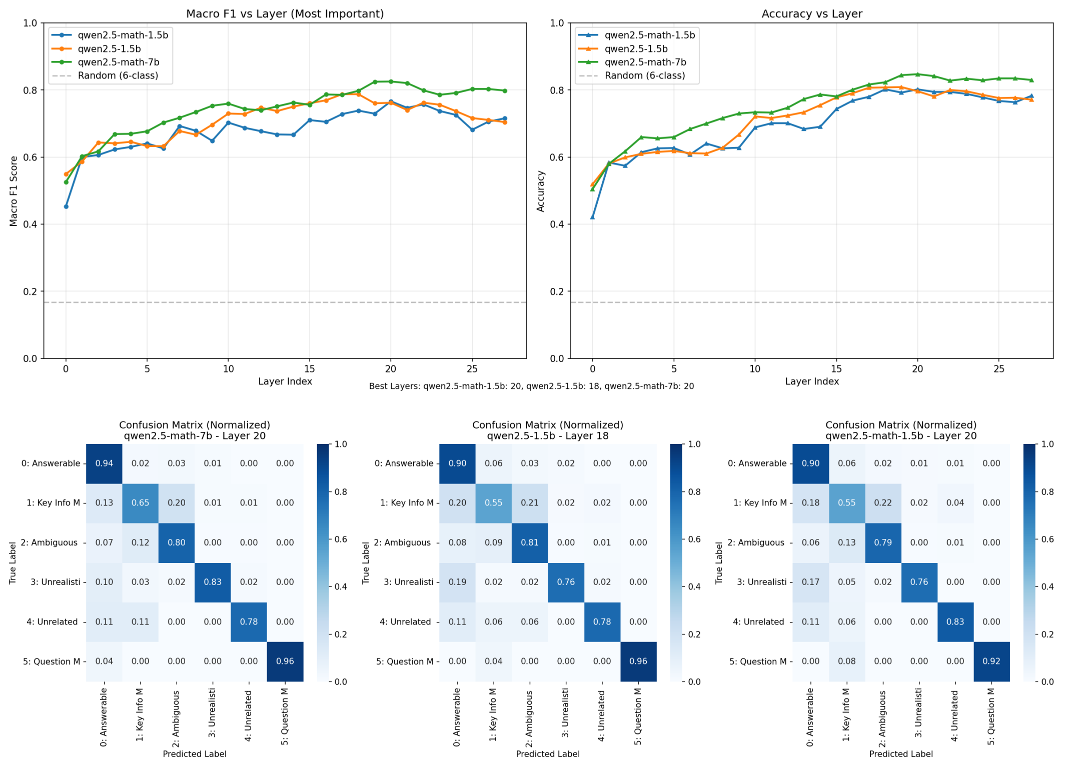


Figure 4: Fine-grained multi-class insufficiency classification on UMWP (6 classes: answerable + 5 insufficiency types). **Top row**: Layer-wise macro F1 and accuracy across all four models. Performance peaks at layers 18–20 (77.9–82.5% F1), indicating that fine-grained insufficiency awareness emerges in late layers. **Bottom row**: Normalized confusion matrices at best-performing layer per model. Models clearly distinguish answerable problems (Class 0) and structural failures (Class 5: Question Missing), while showing expected overlap between semantically similar classes (Class 1: Key Info Missing and Class 2: Ambiguous Info). Unrealistic Conditions (Class 3) forms a distinct cluster, reflecting its unique logical contradiction structure.

4.3 Fine-Grained Multi-Class Classification

Having demonstrated that binary insufficiency is robustly decodable (§4.1) and generalizable (§4.2), we now investigate a deeper question: **do models internally represent *what* information is missing, or merely *that* something is missing?** Logical insufficiency manifests in multiple forms beyond simple numerical omissions—problems may contain contradictory constraints, inappropriate question types, or structural failures [28]. To test whether models encode these fine-grained distinctions, we extend our binary probes to 6-class classification on UMWP's structured insufficiency taxonomy.[13]

Figure 4 shows layer-wise performance (top row) and normalized confusion matrices at the best-performing layer (bottom row) for all four models. Performance peaks at layers 18–20, achieving macro F1 scores of 77.9–82.5% and accuracy of 75.8–84.7%, significantly outperforming the random baseline (16.7%). This confirms that **insufficiency types are linearly separable in late layers**, with models explicitly allocating subspace structure to distinguish semantic versus structural failures.

Structural vs. Semantic Insufficiency. The confusion matrices reveal a clear hierarchy in detection difficulty that aligns with the cognitive distinction between structural and semantic failures. **Question Missing** (Class 5)—a structural failure where the problem lacks a coherent question—is detected with near-perfect precision across all models (F1 0.92–0.98). This suggests that models recognize

[13]UMWP provides five insufficiency types: (1) Key Info Missing (essential numerical values omitted), (2) Ambiguous Info (underspecified relationships), (3) Unrealistic Conditions (logical contradictions), (4) Unrelated Object (question asks for information not derivable from context), and (5) Question Missing (structural failure where the question itself is absent). Class 0 is answerable/sufficient.

syntactic completeness as a distinct, easily separable property [11]. In contrast, **Key Info Missing** (Class 1)—a semantic failure requiring logical dependency tracking to identify which numerical value is absent—is consistently the hardest to classify (F1 0.59–0.70 for 1.5B models, 0.70 for 7B model). The performance gap between structural and semantic detection persists even at 7B scale, indicating that fine-grained semantic reasoning about information flow remains challenging even when coarse-grained sufficiency detection succeeds.

Geometric Overlap Between Semantic Classes. The primary source of classification errors occurs between semantically related insufficiency types. **Key Info Missing** (Class 1) and **Ambiguous Info** (Class 2) exhibit substantial confusion: models correctly classify Class 1 only 59–65% of the time, misclassifying it as Class 2 in 15–20% of cases. This overlap is geometrically interpretable—both classes involve missing or underspecified information, differing primarily in whether a concrete value is absent (Class 1) versus a relationship being ambiguous (Class 2). The boundary between these categories is genuinely fuzzy even for human annotators [28], and models appear to learn this natural ambiguity rather than imposing an artificial hard decision boundary.

Conversely, **Unrealistic Conditions** (Class 3)—problems containing logical contradictions rather than missing data—forms a relatively distinct cluster (F1 0.79–0.85). This suggests that contradictions occupy a separate region of representation space from information-absence patterns, consistent with models learning compositionally structured insufficiency representations rather than entangled features [37].

These fine-grained results demonstrate that models encode not only the binary property of insufficiency but also a structured hierarchy of insufficiency types. The geometric separability of structural failures, the confusion between semantically adjacent classes, and the distinct clustering of contradiction-based insufficiency all suggest that **models internally represent a compositional taxonomy of what information is missing**, even when they cannot verbally articulate these distinctions (§4.2).

4.4 Training-Free Intervention

Having established that models internally encode insufficiency as a robust, linearly separable signal (§4.1–4.3), we now evaluate whether these latent representations can guide practical intervention. Our central question: **can probe-detected insufficiency be leveraged at inference time to elicit acknowledgment and identification of missing information, without fine-tuning or auxiliary models?**

Figure 1 illustrates our training-free intervention approach. For each insufficiency-detection query: (1) we extract frozen representations and apply a trained linear probe to predict insufficiency, (2) if the probe detects insufficiency (prediction = 1), we prepend a system prompt warning the model that information may be missing and explicitly request identification of what is needed,[14] (3) the model generates a response analyzing the problem, and (4) we employ GPT-4o-mini as a judge to evaluate whether the model (a) acknowledged insufficiency and (b) correctly identified the missing information.

This pipeline tests two hypotheses. First, **detection hypothesis**: probes trained on best-performing layers (§4.1) will maintain high detection rates on held-out insufficient questions. Second, **verbalization hypothesis**: explicit probe-guided warnings can surface latent insufficiency knowledge, enabling models to articulate what they internally represent but fail to express under standard zero-shot prompting (§4.2).

Due to computational constraints, we evaluate two representative models (Qwen2.5-Math-1.5B, Qwen2.5-1.5B-Instruct) on two datasets (UMWP, GSM8K-Insufficient). For each model-dataset pair, we train a linear probe on the best-performing layer identified in §4.1 using the full training set, then test on all insufficient questions from the held-out test set.[15] The judge evaluates acknowledgment

[14]The system prompt informs the model: "The following question might be missing some information needed to solve it completely. If information is missing, clearly identify what specific information you would need to provide a definitive answer. Do not make assumptions." Complete intervention prompts are provided in Appendix F.

[15]We use layers 21 (Qwen2.5-Math-1.5B on UMWP), 18 (both models on GSM8K), and 17 (Qwen2.5-1.5B on UMWP), matching the layer-wise analysis in Table 3. Training takes 0.35–2.5 seconds per probe (Appendix G).

Table 5: Training-free intervention results on insufficient test questions. Probe detection measures the proportion of insufficient questions correctly identified by the linear probe. Acknowledgment rate (of detected) measures models' recognition of insufficiency when explicitly warned. Identification rate (given ack.) measures correct pinpointing of missing information among acknowledged cases. Identification rate (overall) measures end-to-end success: detected by probe, acknowledged, and correctly identified. Timing shows average per-question latency. Due to computational constraints, we evaluate two representative models on two datasets.

Model	**Dataset**	**Probe Detect**	**Ack. Rate**	**ID (given ack.)**	**ID (overall)**	**Probe Time**	**Model Time**
Qwen2.5-Math-1.5B	UMWP	89.4%	84.1%	77.9%	67.9%	0.8ms	25.8s
	GSM8K	90.2%	96.4%	82.4%	79.9%	0.8ms	24.0s
Qwen2.5-1.5B	UMWP	88.3%	87.8%	59.4%	54.0%	0.9ms	16.8s
	GSM8K	89.8%	93.3%	68.9%	65.3%	0.8ms	16.8s

(YES if model recognizes insufficiency, NO if model proceeds with assumptions) and correct identification (YES if model identifies the specific missing information, NO if vague or incorrect, N/A if not acknowledged). Evaluation details are provided in Appendix F.

Detection and Intervention Results. Table 5 reports comprehensive intervention metrics. Probe detection rates remain high (88.3–90.2%), confirming that linear separability holds on held-out test questions. Among probe-detected cases, acknowledgment rates are remarkably strong (84.1–96.4%), demonstrating that **explicit insufficiency warnings successfully trigger epistemic humility**. Models no longer silently assume missing constraints; instead, they recognize and articulate uncertainty when guided by probe predictions.

Correct identification rates—the proportion of acknowledged cases where models pinpoint the specific missing information—vary by model capability and dataset complexity. The math-specialized Qwen2.5-Math-1.5B achieves 77.9–82.4% identification accuracy (given acknowledgment), substantially outperforming the general-purpose Qwen2.5-1.5B (59.4–68.9%). This suggests that domain-specific pretraining not only improves solving ability (Table 2) but also enhances models' capacity to reason about information dependencies [35]. Identification performance is higher on GSM8K than UMWP across both models, likely reflecting GSM8K's simpler insufficiency structure (single missing numerical value) versus UMWP's diverse taxonomy including structural contradictions and inappropriate question types (§4.3).

Probe inference is remarkably fast (<1ms per question), adding negligible overhead compared to model generation (16.8–25.8s depending on model size). This confirms that probe-guided intervention is **practically deployable at inference time without auxiliary models or post-hoc verification** [18, 8]. The entire pipeline (probe detection + guided generation + GPT-4o-mini judgment) completes in under 30 seconds per question, making it suitable for interactive applications where preventing hallucination before generation is preferable to costly multi-sample verification [7].

These results validate our central thesis: models already encode insufficiency awareness in frozen representations, and lightweight linear probes can surface this latent knowledge to guide verbalization. By detecting insufficiency *before* generation and explicitly prompting for clarification, we achieve training-free epistemic humility without modifying model weights—a practical pathway from interpretability insights to deployment [36, 31].

5 Discussion

Our findings reveal a fundamental disconnect between what LLMs internally represent and what they verbally express. Models encode logical insufficiency as a robust, linearly separable signal achieving 80–94% F1 on binary detection, yet their zero-shot verbal self-assessment hovers near chance. This +25.6pp representation-language gap persists across model scales (1.5B–7B) and datasets, confirming that overconfident hallucination on insufficient questions reflects a *verbalization failure* rather than a recognition failure. Lightweight probe-guided intervention successfully bridges this gap: when explicitly warned based on probe predictions, models acknowledge insufficiency in 84–96% of detected cases and correctly identify missing information in 59–82% of acknowledged cases, without fine-tuning or auxiliary verification.

These results have practical implications for deployment. Current hallucination mitigation relies on post-hoc verification [18, 20] or multi-sample consistency [7], incurring 10–100× computational overhead. Our approach offers a training-free alternative: detect insufficiency via sub-millisecond probe inference, then guide generation with targeted prompts. This prevents hallucination before it occurs, aligning with recent work on representation engineering [36] and activation steering [31] that leverages internal representations to modify behavior without weight updates. Limitations include restriction to mathematical reasoning (generalization to open-domain QA remains untested) and reliance on GPT-4o-mini for evaluation (human validation would strengthen claims). Future work should explore whether probe-guided interventions transfer to factual hallucination, whether fine-tuning on probe-identified cases improves verbalization, and whether activation steering on insufficiency directions can modify behavior more directly than prompting.

6 Conclusion

We investigated whether LLMs internally recognize logical insufficiency in mathematical reasoning problems, despite their tendency to confidently hallucinate on such questions. Through systematic probing experiments across three benchmarks and four models, we demonstrate that insufficiency is robustly encoded as a linearly separable property in frozen representations (80–94% F1), generalizes across datasets (80–92% F1 on natural problems), and extends to fine-grained insufficiency type classification (78–83% F1 on 6 classes). Yet models' verbal self-assessment remains near chance, revealing a +25.6pp representation-language gap. We exploit this gap via training-free probe-guided intervention: detecting insufficiency in sub-millisecond latency and prompting models to articulate missing information before generation. Models acknowledge insufficiency in 84–96% of probe-detected cases and correctly identify missing constraints in 59–82% of acknowledged cases, offering a practical pathway to epistemic humility without fine-tuning. These findings confirm that models already "know" when they cannot know—the challenge lies in surfacing this latent knowledge at inference time.

References

[1] Meta AI. Llama 3.2: Compact multilingual large language models for diverse applications. `https://huggingface.co/meta-llama/Llama-3.2-3B-Instruct`, 2024.

[2] Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes. In *International Conference on Learning Representations Workshop*, 2017.

[3] Amos Azaria and Tom Mitchell. The internal state of an llm knows when it's lying. *arXiv preprint arXiv:2304.13734*, 2023.

[4] Yoav Bakman, Sungmin Kang, Zhiqi Huang, Deniz Yaldiz, Catarina G Belém, Chenyang Zhu, Anoop Kumar, Alfy Samuel, Salman Avestimehr, Daben Liu, et al. Uncertainty as feature gaps: Epistemic uncertainty quantification of LLMs in contextual question-answering. *arXiv preprint arXiv:2510.02671*, 2025.

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

[6] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[7] Chao Chen, Kai Gao, Rong Xu, Kai Zhang, Fuzheng Chen, and Zuozhu Chao. INSIDE: Llms' internal states retain the power of hallucination detection. In *The Twelfth International Conference on Learning Representations*, 2024.

[8] I-Chun Chern, Steffi Chern, Shiqi Chen, Weizhe Guo, Pengfei Qian, Junda Liu, Yue Gu, Pingzhi Zhou, Hung-Yi Lee, Wenhu Chen, et al. FacTool: Factuality detection in generative AI—a tool augmented framework for multi-task and multi-domain scenarios. *arXiv preprint arXiv:2307.13528*, 2023.

[9] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[10] Owain Evans, Owen Cotton-Barratt, Lukas Finnveden, Adam Bales, Avital Balwit, Peter Wills, Luca Righetti, and William Saunders. Truthful AI: Developing and governing AI that does not lie. *arXiv preprint arXiv:2110.06674*, 2021.

[11] Mor Geva, Jasmijn Bastings, Katja Filippova, and Amir Globerson. Dissecting recall of factual associations in auto-regressive language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12216–12235, 2023.

[12] 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*, 2021.

[13] Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38, 2023.

[14] Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, et al. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*, 2022.

[15] Boaz Lavi and Tomer Milo. Detecting (un)answerability in large language models with probes. *arXiv preprint arXiv:2404.12534*, 2024.

[16] Rui Li, Jing Long, Muge Qi, Heming Xia, Lei Sha, Peiyi Wang, and Zhifang Sui. Towards harmonized uncertainty estimation for large language models. In *Annual Meeting of the Association for Computational Linguistics*, 2025.

[17] Stephanie Lin, Jacob Hilton, and Owain Evans. Teaching models to express their uncertainty in words. *Transactions on Machine Learning Research*, 2024.

[18] Potsawee Manakul, Adian Liusie, and Mark JF Gales. SelfCheckGPT: Zero-resource black-box hallucination detection for generative large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9004–9017, 2023.

[19] Samuel Marks and Max Tegmark. The geometry of truth: Emergent linear structure in large language model representations of true/false datasets. *arXiv preprint arXiv:2310.06824*, 2023.

[20] Sewon Min, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen-tau Yih, Pang Wei Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. FActScore: Fine-grained atomic evaluation of factual precision in long form text generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12076–12100, 2023.

[21] Jialin Ouyang. Treecut: A synthetic unanswerable math word problem dataset for llm hallucination evaluation. *arXiv preprint arXiv:2502.13442*, 2025.

[22] Baolin Peng, Michel Galley, Pengcheng He, Hao Cheng, Yujia Xie, Yu Hu, Qiuyuan Huang, Lars Liden, Zhou Yu, Weizhu Chen, et al. Check your facts and try again: Improving large language models with external knowledge and automated feedback. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 1–22, 2023.

[23] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[24] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, 2018.

[25] Aviv Slobodkin, Omer Goldman, Avi Caciularu, Ido Dagan, and Roi Reichart. The curious case of hallucinatory (un)answerability: Finding truths in the hidden states of over-confident large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 1–20, 2024.

[26] Gaurang Sriramanan, Shaurya Uppal, Mengyu Zhang, Rajiv Mathews, Yi Wu, and Limin Tong. LLM-Check: Investigating detection of hallucinations in large language models. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

[27] Weihang Su, Changyue Wang, Qingyao Ai, Yiran Hu, Zhijing Wu, Yujia Zhou, and Yiqun Liu. Unsupervised real-time hallucination detection based on the internal states of large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 14262–14278, 2024.

[28] Yuhong Sun, Zhangyue Yin, Qipeng Guo, Jiawen Wu, Xipeng Qiu, and Hui Zhao. Benchmarking hallucination in large language models based on unanswerable math word problem. *arXiv preprint arXiv:2403.03558*, 2024.

[29] Qwen Team. Qwen2.5: A party of foundation models. `https://qwenlm.github.io/blog/qwen2.5/`, September 2024.

[30] Curt Tigges, Oskar John Hollinworth, Atticus Geiger, and Neel Nanda. Linear representations of sentiment in large language models. *arXiv preprint arXiv:2310.15154*, 2023.

[31] Alexander Matt Turner, Lisa Thiergart, David Udell, Gavin Leech, Ulisse Mini Kenton, Juan Felipe Ceron de Oliveira Marinho, and Alex Gerovitch. Steering language models with activation engineering. *arXiv preprint arXiv:2308.10248*, 2024.

[32] Alexander Matt Turner, Lisa Thiergart, David Udell, Gavin Leech, Ulisse Mini, and Monte Kenton. Activation addition: Steering language models without optimization. *arXiv preprint arXiv:2308.10248*, 2024.

[33] Jian Xie, Kai Zhang, Jiangjie Chen, Renze Meng, Chenjuan Lou, Cheng Guo, and Yanghua Zhang. Adaptive chameleon or stubborn sloth: Unraveling the behavior of large language models in knowledge conflicts. *arXiv preprint arXiv:2305.13300*, 2023.

[34] Miao Xiong, Zhiyuan Hu, Xinyang Lu, Yifei Li, Jie Fu, Junxian He, and Bryan Hooi. Can LLMs express their uncertainty? an empirical evaluation of confidence elicitation in LLMs. *arXiv preprint arXiv:2306.13063*, 2023.

[35] An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*, 2024.

[36] Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, et al. Representation engineering: A top-down approach to AI transparency. *arXiv preprint arXiv:2310.01405*, 2023.

[37] Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, et al. Representation engineering: A top-down approach to AI transparency. *arXiv preprint arXiv:2310.01405*, 2023.

A GSM8K Insufficient Transformation Prompt

We use the following prompt to programmatically generate insufficient variants from GSM8K problems via GPT-4o. The prompt emphasizes naturalness, grammatical correctness, and removal of exactly one critical numerical value to ensure genuine logical insufficiency.

GPT-4o Transformation Prompt

```
You are a research assistant helping to create a dataset for studying
logical insufficiency in mathematical reasoning problems.

DEFINITION OF LOGICAL INSUFFICIENCY:
A math problem is "logically insufficient" when it lacks a critical piece of
information needed to arrive at a unique, definitive answer. This is NOT about
ambiguous language or unclear questions - it's about missing concrete numerical
values or constraints that are mathematically necessary to solve the problem.

ORIGINAL QUESTION:
{question}

YOUR TASK:
Transform this question into a logically insufficient version by removing
EXACTLY ONE critical numerical value.

DETAILED REQUIREMENTS:

1. WHAT TO REMOVE:
   - Remove ONE number that is essential for solving.
   - The removed number should be one that, without it, multiple valid answers
   become possible.
   - Choose a number whose absence makes the problem genuinely unsolvable,
   not just harder.

2. HOW TO REPLACE IT:
   - Replace with natural language like:
     * "some" (e.g., "12 tickets" → "some tickets")
     * "several" (e.g., "15 students" → "several students")
     * "a certain amount" (e.g., "$30" → "a certain amount")
   - The replacement should sound natural in context
   - Do NOT use phrases like "X" or "an unknown value"

3. PRESERVE EVERYTHING ELSE:
   - Keep ALL other numbers unchanged
   - Keep ALL other text EXACTLY as written
   - Maintain perfect grammatical correctness

OUTPUT FORMAT:
Return ONLY a valid JSON object with these exact fields:
{
  "insufficient_question": "the modified question text",
  "removed_value": "the exact number you removed as a string",
  "removed_description": "brief description of what this
                          value represents"
}
```

B GPT-4o-mini Answer Judging Protocol

For evaluating model accuracy on sufficient questions (Table 2), we employ GPT-4o-mini as an automated judge to compare model-generated answers against ground-truth solutions. The judge extracts the final numerical answer from model outputs (looking for common patterns like `####` `<number>`, `\boxed{<number>}`, or final numbers), then performs strict numerical equality checking within floating-point precision ($\text{abs}(\text{model\_answer} - \text{ground\_truth}) < 10^{-10}$). Below is the complete judging prompt.

GPT-4o-mini Answer Judging Prompt

```
You are a mathematics answer extraction and evaluation expert for
academic research.

## TASK
Extract the final numerical answer from a model's response and compare
it to the ground truth answer.

## EXTRACTION RULES

1. Look for common answer patterns (in order of priority):
   - '#### <number>' (e.g., "#### 72")
   - '\boxed{<number>}' (e.g., "\boxed{72}")
   - "the answer is <number>" or "therefore <number>"
   - Final number at the end of the response

2. If multiple candidate answers appear: Take the LAST one (final answer)
3. Ignore intermediate calculations: Only extract the final result
4. Extract ONLY the number: No units, no text, no formatting
5. If no clear numerical answer found → verdict: "incorrect"

## COMPARISON RULES

1. Parse both values as numbers (float comparison)
2. STRICT NUMERICAL EQUALITY: abs(model_answer - ground_truth) < 1e-10
3. No tolerance: We require exact equality (within floating point
   precision)

## OUTPUT FORMAT

You MUST respond with valid JSON only:

{
  "verdict": "correct" | "incorrect",
  "model_answer": "<extracted number as string>" | null,
  "ground_truth": "<ground truth number as string>",
  "explanation": "<brief one-sentence reason for verdict>"
}

## YOUR TASK

Question: {question}
Ground Truth: {ground_truth}
Model Output: {model_output}

Evaluate the model's answer and respond with JSON only.
```

This protocol ensures robust evaluation across diverse output formats while maintaining strict numerical correctness standards.

C Verbal Self-Assessment Prompts

To evaluate the representation-language gap (§4.2), we measure models' ability to verbally express insufficiency awareness through zero-shot prompting. This involves two stages: (1) prompting the LLM to assess whether a problem can be solved, and (2) using GPT-4o-mini to parse and classify the LLM's response. Below are the exact prompts used for both stages.

C.1 LLM Self-Assessment Prompt

We prompt each model with the following zero-shot instruction, requesting structured output via `\boxed{}` notation to facilitate reliable parsing:

LLM Self-Assessment Prompt

```
Can this math problem be solved with the given information?
Answer Yes or No and put your answer in \boxed{}.

Problem: {question}

Answer:
```

where `{question}` is replaced with the problem text. We extract the model's response and pass it to a GPT-4o-mini judge for classification.

C.2 GPT-4o-mini Binary Assessment Judge

To robustly classify model responses that may contain extraneous text, hedging, or non-standard formatting, we employ GPT-4o-mini as a binary assessment judge. The judge first attempts regex-based Yes/No extraction; if that fails, it uses the following LLM-based classification prompt:

GPT-4o-mini Binary Assessment Judge Prompt

```
You are evaluating whether a model correctly assessed if a math
problem can be solved.

The model was asked: "Can this math problem be solved with the
given information?"

The model should answer "Yes" if sufficient information is provided,
"No" if insufficient.

Evaluate if the model's assessment matches the ground truth.

Return JSON with:
- "verdict": "correct" or "incorrect"
- "model_answer": extracted Yes/No from model output
- "ground_truth": the correct Yes/No answer
- "explanation": brief reasoning

## YOUR TASK

Question: {question}
Ground Truth Answer: {ground_truth}
Model Output: {model_output}

Please evaluate whether the model's assessment is correct.
Respond with JSON only.
```

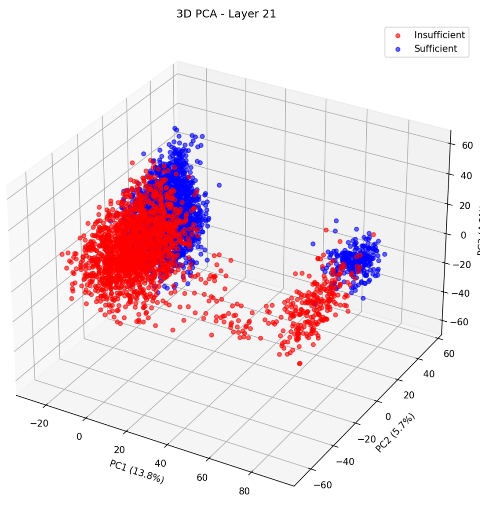
where `{question}` is the problem, `{ground_truth}` is "Yes" (sufficient) or "No" (insufficient), and `{model_output}` is the LLM's raw response. The judge returns a verdict (correct/incorrect) which we use to compute verbal self-assessment accuracy. This two-stage pipeline (regex extraction with LLM fallback) ensures robust evaluation even when models produce verbose or ambiguous responses.

D Additional Mechanistic Interpretability Analysis

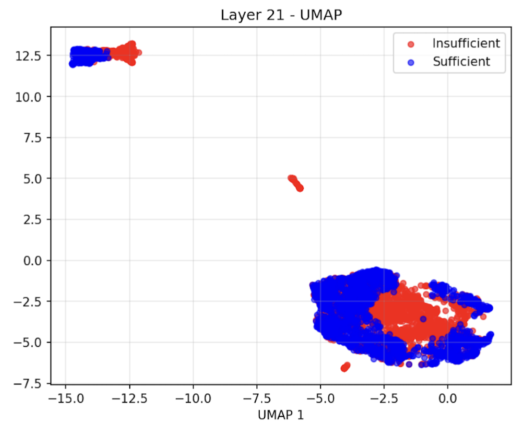
This appendix provides detailed visualizations of the geometric and algebraic structure of insufficiency representations in Qwen2.5-Math-1.5B and Llama-3.2-1B-Instruct, complementing the summary in §4.1.

D.1 Geometric Visualization: PCA and UMAP

To visualize the representational geometry of insufficiency, we apply PCA and UMAP dimensionality reduction to hidden states at layer 21 (the best-performing layer for Qwen2.5-Math-1.5B). Figure 5 shows both visualizations. The 3D PCA projection (left) reveals that the two classes (sufficient in blue, insufficient in red) form clearly separated clusters with minimal overlap. The first three principal components capture 26.5% of variance and already exhibit strong class separation. The UMAP embedding (right) confirms this geometric structure with two distinct, compact clusters achieving near-perfect separation. These visualizations demonstrate that insufficiency is encoded as a structurally salient property in late-layer representations, not as a subtle perturbation buried in high-dimensional noise.



(a) 3D PCA (Layer 21)



(b) UMAP Visualization (Layer 21)

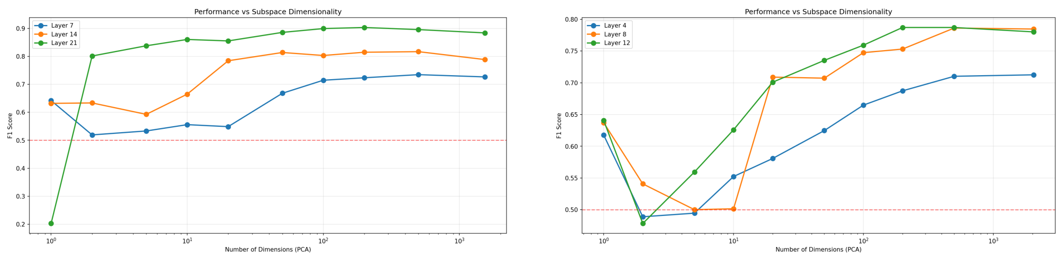
Figure 5: Geometric visualization of insufficiency representations in Qwen2.5-Math-1.5B at layer 21 (best-performing layer). **Left**: 3D PCA shows clear cluster separation with the first three components capturing 26.5% of variance. **Right**: UMAP reveals two distinct, compact clusters with minimal overlap. Both visualizations confirm that insufficiency is encoded as a geometrically distinct property.

D.2 Subspace Dimensionality Analysis

To quantify the intrinsic dimensionality of insufficiency representations, we train probes on PCA-reduced embeddings with varying numbers of components (1, 2, 5, 10, 20, 50, 100, 200, 500). Figure 6 plots F1 score versus number of PCA components (log scale) for three representative layers per model.

For late layers (21 in Qwen2.5-Math-1.5B, 12 in Llama-3.2-1B-Instruct), performance plateaus at 100–200 dimensions, achieving $>$85% of full-dimensional F1—a compression ratio of $>10\times$ (from 1536/2048 to $\sim$100 dimensions). Remarkably, even with just 1 PCA component (the single direction of maximum variance), late-layer probes achieve 60–80% F1. This low-rank structure confirms that models explicitly allocate a compact subspace for insufficiency awareness rather than entangling it with high-dimensional task features [19, 3].

These findings demonstrate that insufficiency is not a subtle perturbation buried in high-dimensional noise, but rather an explicit, low-dimensional signal that emerges and strengthens across network



(a) Qwen2.5-Math-1.5B (b) Llama-3.2-1B-Instruct

Figure 6: Subspace dimensionality analysis. F1 score vs. number of PCA components for selected layers. Late layers (green) saturate at 100–200 dimensions, achieving >85% of full performance. Even a single direction captures 60–80% F1, confirming low-rank structure.

depth. The geometric separability and low-rank structure make insufficiency representations particularly amenable to lightweight intervention mechanisms (§4.4).

E Cross-Dataset Generalization with Mean Pooling

For completeness, we report cross-dataset generalization results using mean pooling aggregation (averaging hidden states across all token positions) rather than last token pooling. Table 6 shows F1 scores for all four models.

Mean pooling exhibits qualitatively similar transfer patterns to last token pooling (Table 3 in main text): strong generalization between UMWP and GSM8K (77–93% F1), weaker but substantial transfer to/from TreeCut (51–98% F1), and robust multi-domain probes when trained on ALL datasets (84–93% F1). However, mean pooling consistently achieves slightly degraded absolute performance compared to last token pooling, particularly on TreeCut where the gap is most pronounced (e.g., Qwen2.5-Math-1.5B: 86.0% vs. 76.0% F1 for TreeCut in-distribution). This confirms that **last token pooling is the superior aggregation strategy** for insufficiency detection in decoder-only models, consistent with standard practice in autoregressive language modeling [23, 6].

Table 6: Cross-dataset generalization with mean pooling (F1 scores). Rows indicate training dataset, columns indicate test dataset. Format matches Table 3 for direct comparison. Mean pooling shows similar transfer patterns but slightly degraded absolute performance, confirming last token pooling as the more effective strategy.

Model	Train \ Test	UMWP	GSM8K	TreeCut	ALL
Qwen2.5-Math-1.5B	UMWP	**84.5**	84.3	64.0	85.0
	GSM8K	77.6	**89.9**	66.9	87.0
	TreeCut	65.4	63.6	**86.0**	83.7
	ALL	85.0	87.0	83.7	**85.2**
Qwen2.5-1.5B	UMWP	**85.8**	86.8	70.4	86.1
	GSM8K	80.4	**93.2**	67.4	89.6
	TreeCut	72.5	71.9	**95.4**	90.4
	ALL	86.1	89.6	90.4	**88.7**
Llama-3.2-3B	UMWP	**77.9**	79.6	61.0	79.2
	GSM8K	68.8	**85.5**	60.8	78.4
	TreeCut	61.1	60.0	**81.7**	78.3
	ALL	79.2	78.4	78.3	**78.6**
Qwen2.5-Math-7B	UMWP	**88.4**	87.4	63.7	83.6
	GSM8K	79.7	**92.8**	59.5	85.6
	TreeCut	63.7	67.5	**96.9**	92.7
	ALL	83.6	85.6	92.7	**87.3**

F Intervention Prompts and Evaluation

This appendix details the complete prompts and evaluation protocol used for the training-free intervention experiments (§4.4).

F.1 Intervention System and User Prompts

When the linear probe detects insufficiency (prediction = 1), we reprompt the model with explicit warnings structured as a system message followed by a user message. The complete prompt format is:

Intervention System Prompt

```
System: You are a helpful math assistant. The following question
is missing some information needed to solve it completely. If
information is missing, clearly identify what specific information
you would need to provide a definitive answer. Do not make
assumptions - if you cannot solve it with the given information,
explain what is missing.

User: Question: {question}

Please analyze this question and:
1. Determine if you have all the information needed to solve it
2. If something is missing, identify EXACTLY what information you need
3. Explain why you cannot provide a definitive numerical answer
   without this information
```

where `{question}` is replaced with the insufficient problem text. This prompt explicitly requests identification of missing information rather than assuming values, aiming to surface the latent insufficiency knowledge revealed by probe analysis (§4.2).

F.2 GPT-4o-mini Intervention Evaluation

We employ GPT-4o-mini to judge model responses along two dimensions: (1) **Acknowledgment**—does the model recognize that information is missing? (2) **Correct Identification**—does the model correctly identify what specific information is needed? The judge receives the original sufficient question (for context), the insufficient variant (what the model saw), the removed value (ground truth), and the model's response. It returns structured JSON with three fields: `acknowledged` (YES/NO), `correctly_identified` (YES/NO/N/A), and `explanation`.

The complete judging prompt emphasizes strict evaluation: partial acknowledgment with subsequent assumption-based solving counts as NO acknowledgment, and vague statements like "need more information" without specifying what counts as NO for correct identification. The judge handles vague ground truth (e.g., "Category 2: Incomplete constraint") by accepting any concrete identification of missing information that would help solve the problem. Below is the complete GPT-4o-mini judging prompt with evaluation criteria and examples:

GPT-4o-mini Intervention Judge Prompt

```
You are evaluating whether a model correctly acknowledged and
identified missing information in a logically insufficient math problem.

The model was informed that the question might be logically insufficient
and was asked to identify what information is missing.

## YOUR ROLE

You will be given:
1. Original sufficient question (with all information)
2. Insufficient question (what the model saw, with something removed)
3. What was removed (ground truth - the specific value/constraint that
   makes the question insufficient)
4. Model's response (how the model reacted to the insufficiency warning)

Your task is to evaluate TWO things:

## EVALUATION CRITERION 1: ACKNOWLEDGMENT

Question: Does the model acknowledge that information is missing?

Answer "YES" if:
- Model explicitly states something is missing, insufficient, or cannot
  be determined

Answer "NO" if:
- Model ignores the insufficiency warning
- Model says "assuming X..." and then solves

## EVALUATION CRITERION 2: CORRECT IDENTIFICATION

Question: If the model acknowledged insufficiency, does it correctly
identify what's missing?

Answer "YES" if:
- Model identifies specific missing information that would allow
  solving the problem

Answer "NO" if:
- Model acknowledges insufficiency but identifies the WRONG missing
  information

Answer "N/A" if:
- Model didn't acknowledge insufficiency at all (answered "NO" to
  Criterion 1)

## OUTPUT FORMAT

You MUST respond with valid JSON only (no markdown, no code blocks):

{
  "acknowledged": "YES" | "NO",
  "correctly_identified": "YES" | "NO" | "N/A",
  "explanation": "<brief one-sentence reasoning for your judgment>"
}

Evaluate the model's response and respond with JSON only.
```

This prompt includes detailed evaluation criteria and examples to ensure consistent judging across all intervention experiments.

G Probe Training Computational Cost

All linear probes used in this work are trained via scikit-learn's logistic regression with LBFGS optimizer, maximum 7,000 iterations, and $L2$ regularization strength $\lambda = 1.0$. Training is performed on frozen representations extracted once and cached to disk, eliminating redundant forward passes through the LLM.

Table 7 reports training time, training F1, and best-performing layer for all model-dataset pairs evaluated in the intervention experiments (§4.4). Probe training completes in 0.35–2.5 seconds depending on training set size (1,632 examples for GSM8K, 4,160 for UMWP), achieving near-perfect training F1 (0.99–1.00) across all configurations. Inference latency is sub-millisecond ($<$1ms per question), confirming that probe-guided intervention adds negligible computational overhead compared to model generation (16.8–25.8s per question). This extreme efficiency—training in seconds, inference in microseconds—makes linear probes a practical tool for deployment without auxiliary models or post-hoc verification.

Table 7: Probe training computational cost for intervention experiments. All probes achieve near-perfect training F1 in under 3 seconds, with inference latency under 1 millisecond. Best layers are identified via layer-wise search (§4.1).

Model	**Dataset**	**Train Size**	**Best Layer**	**Train Time**	**Train F1**
Qwen2.5-Math-1.5B	UMWP	4,160	21	2.5s	1.00
Qwen2.5-Math-1.5B	GSM8K	1,632	18	0.9s	0.99
Qwen2.5-1.5B	UMWP	4,160	17	2.3s	1.00
Qwen2.5-1.5B	GSM8K	1,632	18	0.35s	1.00

The combination of fast training, perfect training fit, and strong test generalization (Table 5) confirms that insufficiency is a robustly linearly separable property that can be extracted via lightweight probes without overfitting or computational burden.