Your Reasoning Model is Secretly a Reward Model - Training-Free Verification from Experience

Anonymous ACL submission

Abstract

Assessing the quality of Large Language Model (LLM) outputs becomes especially challenging in high-branching settings, where a single prompt yields many plausible candidates. Existing verifiers typically operate on the surface text (e.g., reward models, LLM judges, majority voting) or on confidence proxies derived from token probabilities, both of which can be brittle: the former can be influenced by stylistic artifacts, while the latter is often miscalibrated. In this paper, we study a third source of information—the model's hidden states—for *binary correctness verification* in tasks with a reliable success/failure signal (e.g., deterministic checkers or reference-grounded answers). We find that correct and incorrect solutions exhibit measurable geometric differences in their hidden-state trajectories. To isolate this signal with minimal modeling assumptions, we introduce **CLUE (Clustering and Experience-based Verification)**, a training-free, non-parametric verifier. CLUE summarizes each reasoning trace by an *activation delta*—the difference between hidden states at the start and end of the explicit reasoning span—and predicts correctness by comparing this delta to two class centroids computed from labeled experience. Across math (AIME 24/25), scientific QA (GPQA), and a multi-domain benchmark (WebInstruct-verified), CLUE improves selection and reranking, with particularly strong gains on smaller or less-calibrated models. For example, on AIME 24 with a 1.5B model, CLUE raises accuracy from 56.7% (majority@64) to 70.0% (top-maj@16).

1 Introduction

Large Language Models (LLMs) can generate many candidate solutions for a single prompt, especially on challenging reasoning tasks. This capability shifts the bottleneck from generation to verification (Cobbe et al., 2021; Lightman et al., 2023; Hosseini et al., 2024): when dozens of plausible-looking answers are available, how can we reliably select the correct one among convincing but incorrect alternatives?

To address this question, the research community has largely pursued two main strategies. The first operates purely on the surface of the generated text, delegating evaluation to an external judge. This includes training separate reward models (Ouyang et al., 2022; Bai et al., 2022; Zheng et al., 2023) or adopting simple heuristics such as majority voting (Wang et al., 2022). While useful in practice, these after-the-fact approaches are fundamentally blind to the model's actual reasoning process. They can be systematically misled by stylistic artifacts—e.g., verbose but incorrect answers often receive higher scores than terse but correct ones (Glaese et al., 2022). Moreover, trained judges inherit biases and limitations from their training data, making them brittle under distribution shift and expensive to retrain for new domains.

A second line of work attempts to go beneath the surface, but only through the shadow of the model's reported confidence. Methods in this category rely on token probabilities, entropy, or derived uncertainty estimates (Kadavath et al., 2022b; Lin et al., 2023; Geng et al., 2023; Xiong et al., 2024; Fu et al., 2025b). The underlying assumption is that higher probability correlates with higher correctness. However, calibration of LLMs remains poor: even state-of-the-art models are often "confidently wrong," assigning extreme likelihood to factually false or logically inconsistent outputs (Fu et al., 2025a). These confidence-based metrics also degrade significantly on smaller or less-tuned models, where probability distributions are noisier and less interpretable, making them a fragile basis for reliable verification.

In this work, we argue that correctness can often be diagnosed more reliably from the model's internal reasoning trajectory than from surface text alone or from token-level confidence proxies. Hid-

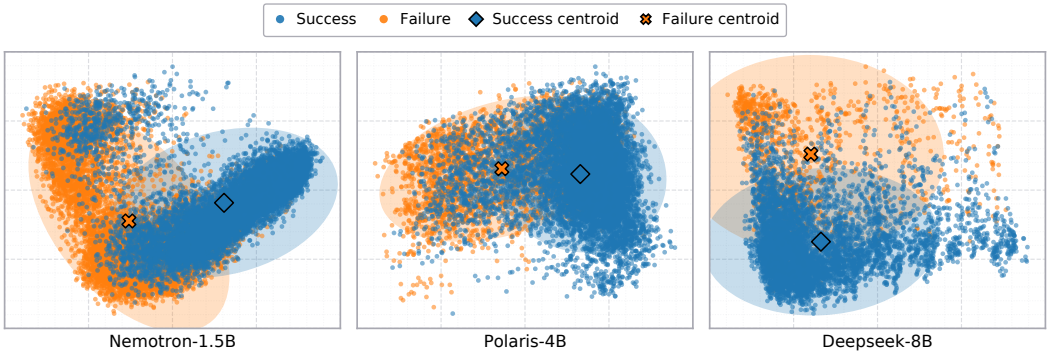

Figure 1: Visualization of hidden-state trajectories for correct (blue) and incorrect (orange) solutions from our experience set, projected to 2D using PCA. Each panel displays a different base model. The separation is imperfect (overlap remains in 2D), but a consistent geometric shift is visible across models, motivating a simple centroid-based verifier.

den states contain information relevant to both: earlier layers are closer to token embeddings and reflect semantic/lexical features, while later layers align more strongly with the output logits and correlate with probability-based cues. This motivates hidden states as a unified representation for verification.

Our core hypothesis is a *geometric* one: conditioned on a verification signal that cleanly partitions outcomes into success/failure, trajectories that end in correct answers tend to concentrate in different regions of representation space than incorrect ones. Importantly, we do not require a perfectly separated manifold; rather, we ask whether the separation is strong enough that a simple non-parametric rule can exploit it. As illustrated in Figure 1, correct and incorrect trajectories exhibit a consistent geometric shift across models, echoing insights from mechanistic interpretability (nostalgebraist, 2020; Belrose et al., 2023; Tomaz et al., 2025) while extending them toward actionable verification.

This structure motivates a lightweight verifier. If correctness correlates with geometry, then even a low-capacity geometric rule should be able to exploit it. We introduce **CLUE (Clustering and Experience-based Verification)**, a training-free, supervised aggregation framework that operates directly on the model's internal computation. Rather than using absolute final states, CLUE summarizes each reasoning trace by an activation delta—the difference between hidden states at the start and end of an explicit reasoning span (in our setup, delimited by `<think> ...</think>`). This delta reduces prompt-specific offsets and emphasizes the net transformation induced by reasoning. From labeled trajectories, CLUE computes success/failure centroids and classifies a new trace by proximity to these centroids (via a layer-averaged Euclidean distance).

Our experiments support this premise. CLUE consistently matches or surpasses representative LLM-as-a-judge and confidence-based baselines, with especially clear gains on smaller or less-calibrated models where probability cues and surface-level judging are unreliable. Because CLUE performs a one-time, deterministic aggregation without gradient training, it avoids many overfitting failure modes of learned verifiers.

Our contributions are summarized as follows:

- We propose a geometric perspective on verification, showing that correctness is encoded as a separable pattern in the hidden-state trajectories of reasoning models.
- We introduce CLUE, a minimalist, training-free verifier that aggregates experience into activation-delta centroids, achieving strong performance without gradient updates or complex prompt engineering.
- We find that models fine-tuned with Reinforcement Learning exhibit sharper internal separation between correct and incorrect reasoning than SFT counterparts, suggesting that training paradigms can materially affect verifier quality.

2 Related Work

2.1 Latent Reasoning and Activation Geometry

LLMs can reason in latent space instead of (or alongside) explicit token chains, via continuous "thought states" fed back into the model or compact hidden "thinking tokens" that compress CoT (Hao

et al., 2024; Shen et al., 2025). Interpretability tools like the logit lens and tuned lens show that intermediate activations progressively align with output distributions, suggesting layer-wise decodable semantics and confidence signals (nostalgebraist, 2020; Belrose et al., 2023). Hidden-state probes can *self-verify* intermediate answers and enable early exit (Zhang et al., 2025), while semantic clustering of hidden rationales can improve robustness (Knappe et al., 2024). Beyond verification, activation directions can monitor or steer model traits (e.g., sycophancy, hallucination) via *persona vectors* (Chen et al., 2025). Also, in-context activation vectors indicate that linear structure in hidden space can be mapped and reused across tasks (Liu et al., 2024). More broadly, recent surveys on representation engineering highlight how linear directions and activation editing provide a general lens on hidden-state geometry in LLMs (Bartoszcze et al., 2025). Unlike these trained probes or steering methods, our verifier CLUE is training-free and purely reads cross-layer activation deltas.

2.2 Test Time Scaling

Recent research has increasingly focused on test-time scaling – techniques that improve model performance by allocating more computation during inference without changing the model's parameters. Parallel approaches (such as self-consistency (Wang et al., 2022) and ensemble "best-of-N" selection (Snell et al., 2024)) generate multiple independent chain-of-thought solutions and then aggregate or vote on the final answer, significantly boosting accuracy on complex tasks. Sequential approaches (such as iterative self-refinement (Madaan et al., 2023), Tree-of-Thoughts search (Yao et al., 2023)) allow the model to think in multiple steps, using intermediate reasoning to inform subsequent generations. Variants like weighted or semantic self-consistency (Luo et al., 2024; Knappe et al., 2024) highlight the importance of aggregating diverse rationales, while RLHF and LLM-as-a-judge approaches (Ouyang et al., 2022; Zheng et al., 2023) provide external supervision but can be costly and biased. To reduce dependence on large reward models, SWIFT learns *lightweight* hidden-state rewards that scale efficiently to best-of-*N* sampling (Guo et al., 2025b). Complementary to this, DeepConf filters low-quality reasoning traces using internal confidence signals, improving both efficiency and accuracy (Fu et al., 2025b); relatedly, early-exit schemes can truncate overthinking while preserving accuracy (Kadavath et al., 2022a; Yang et al., 2025b). In contrast, CLUE introduces no trainable verifier: it computes success/failure centroids once from past experience and reranks by nearest centroid, showing that correctness is geometrically separable in hidden space.

3 The CLUE Framework

We first outline the core intuition behind CLUE. Each time an LLM solves a problem, its internal computation traces a trajectory through a high-dimensional representation space. We hypothesize that trajectories leading to correct solutions differ systematically from those leading to incorrect ones. CLUE captures this difference via a training-free, supervised aggregation over activation deltas: it summarizes each labeled trajectory, builds class centroids from these summaries, and then verifies a new trajectory by proximity to the centroids. This section formalizes the setup and the resulting geometric rule.

3.1 Problem Formulation

Let an LLM be tasked with generating a response R_i for a prompt P. We define a *trajectory* (or *experience*) $T_i = (P, R_i)$, paired with a ground-truth binary label $y_i \in \{0, 1\}$, where $y_i = 1$ denotes a correct solution (success) and $y_i = 0$ denotes an incorrect solution (failure). The goal is to learn a verification function f that maps a new trajectory T_{new} to a prediction $\hat{y}_{\text{new}} = f(T_{\text{new}}) \in \{0, 1\}$. Unlike text-based or probability-based approaches, f operates exclusively on the hidden-state representations generated during the production of R_{new}. The LLM parameters are kept fixed (frozen) throughout both learning and inference.

3.2 CLUE: Verification via Activation-Delta

The central hypothesis is that the *transformation* of internal states during explicit reasoning contains a robust signal of correctness. We capture this transformation with an *activation delta*, defined as the difference between hidden states at the start and the end of the reasoning block. In this paper, the reasoning block is delimited by `<think>` and `</think>`.

Reasoning boundary extraction. Our implementation uses explicit `<think>` boundaries to consistently extract $\mathbf{h}_{\text{start}}$ and $\mathbf{h}_{\text{end}}$. This is a practical choice rather than a conceptual requirement: any

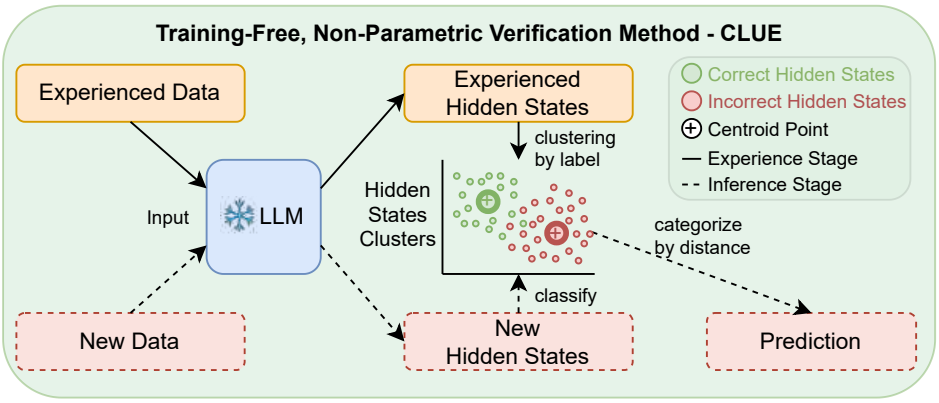

Figure 2: Overview of CLUE. **Left (Learning):** Labeled historical trajectories are summarized by their activation deltas and aggregated into success and failure centroid matrices. **Right (Verification):** A new trajectory is summarized by its activation delta and classified by the layer-averaged Euclidean distance (Eq. 4) to the two pre-computed centroids. The underlying LLM remains *frozen* throughout.

consistent scheme that defines a "reasoning span" (e.g., fixed token offsets, delimiter-free segmentation heuristics, or model-specific conventions) can be used to compute an activation delta. We focus on `<think>` because it is widely adopted by contemporary reasoning models and makes the extraction unambiguous.

Let the model have L layers and hidden dimension D. For a given trajectory T, denote by

$$\mathbf{h}_{\text{start}}(T) \in \mathbb{R}^{L\times D} \quad \text{and} \quad \mathbf{h}_{\text{end}}(T) \in \mathbb{R}^{L\times D}$$

the matrices of hidden states extracted, respectively, at the final token of `<think>` (just before detailed reasoning) and at the final token of `</think>` (after the reasoning has been formed). The activation delta is the matrix

$$\Delta\mathbf{h}(T) = \mathbf{h}_{\text{end}}(T) - \mathbf{h}_{\text{start}}(T) \in \mathbb{R}^{L\times D}. \quad (1)$$

We use hidden states from *all* layers, reflecting the assumption that correctness-related information is distributed across depth (earlier layers retain semantic/lexical cues; later layers align more strongly with logits). The activation-delta matrix $\Delta\mathbf{h}(T)$ serves as the sole feature representation for verification.

Why a delta, and why a nearest-centroid rule? The activation delta in Eq. (1) can be viewed as a coarse summary of the *net representational movement* induced by the model's internal computation during the reasoning span. Subtracting the start state partially cancels prompt-specific offsets that are common to both successful and failed attempts on the same input distribution, emphasizing how the model *updates* its beliefs rather than where it starts. Given labeled experience, a nearest-centroid classifier is the simplest geometric decision rule one can apply: it approximates a class-conditional prototype and is closely related to linear discriminant analysis under spherical covariance. Our goal is not to claim optimality of this classifier, but to use it as a deliberately low-capacity probe. If such a minimalist rule already performs strongly, it indicates that the hidden-state representation itself is highly informative for verification.

3.3 Centroid Construction and Classification

The learning phase is a one-time deterministic statistical aggregation over labeled trajectories $\mathcal{D} = \{(T_i, y_i)\}_{i=1}^{N}$. Define index sets

$$\mathcal{I}_{\text{succ}} = \{\, i \mid y_i = 1 \,\}, \qquad \mathcal{I}_{\text{fail}} = \{\, i \mid y_i = 0 \,\}.$$

For each trajectory, compute its activation delta $\Delta\mathbf{h}_i = \Delta\mathbf{h}(T_i)$ as in Eq. (1). The success and failure *centroid matrices* are the element-wise means:

$$\mathbf{V}_{\text{succ}} = \frac{1}{|\mathcal{I}_{\text{succ}}|} \sum_{i\in\mathcal{I}_{\text{succ}}} \Delta\mathbf{h}_i, \quad (2)$$

$$\mathbf{V}_{\text{fail}} = \frac{1}{|\mathcal{I}_{\text{fail}}|} \sum_{i\in\mathcal{I}_{\text{fail}}} \Delta\mathbf{h}_i. \quad (3)$$

Both $\mathbf{V}_{\text{succ}}, \mathbf{V}_{\text{fail}} \in \mathbb{R}^{L\times D}$ are stored for inference.

At inference, a new trajectory T_{new} is summarized by $\Delta\mathbf{h}_{\text{new}} = \Delta\mathbf{h}(T_{\text{new}})$. We compare it to the two centroids using the *layer-averaged Euclidean distance*. For two matrices $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{L\times D}$ with row vectors $\mathbf{a}_l, \mathbf{b}_l \in \mathbb{R}^D$ (the l-th layer representations), define

$$d(\mathbf{A}, \mathbf{B}) = \frac{1}{L} \sum_{l=1}^{L} \|\mathbf{a}_l - \mathbf{b}_l\|_2. \quad (4)$$

Compute $d_{\text{succ}} = d(\Delta\mathbf{h}_{\text{new}}, \mathbf{V}_{\text{succ}})$ and $d_{\text{fail}} = d(\Delta\mathbf{h}_{\text{new}}, \mathbf{V}_{\text{fail}})$, and classify as

$$\hat{y}_{\text{new}} = \begin{cases} 1, & \text{if } d_{\text{succ}} < d_{\text{fail}}, \\ 0, & \text{otherwise.} \end{cases}$$

This rule matches the high-level description in Figure 2 and requires no gradient-based optimization.

3.4 Application to Solution Reranking

The geometric formulation provides a continuous quality score that is naturally suited for reranking. Given a prompt P and k responses $\{R_1, \ldots, R_k\}$, form trajectories $\{T_1, \ldots, T_k\}$ and their activation deltas $\{\Delta\mathbf{h}_1, \ldots, \Delta\mathbf{h}_k\}$. Define for each candidate

$$s_j = d\big(\Delta\mathbf{h}_j, \mathbf{V}_{\text{succ}}\big), \qquad \text{(lower is better)} \quad (5)$$

and rank candidates in ascending order of s_j. This ranking can be used for top-1 selection or to improve aggregation schemes such as majority vote by prioritizing candidates whose internal reasoning is closest to the success centroid.

3.5 Rationale for a Minimalist, Experience-Based Design

CLUE is intentionally minimalist to isolate the contribution of the representation itself. If a simple, training-free geometric rule over activation-delta summaries yields strong verification performance, this provides evidence that hidden states encode a robust correctness signal. This framing also helps interpret negative results: when CLUE fails, the limitation is often that the underlying representation does not cleanly support correctness separation for that setting, rather than an artifact of optimizer choice or judge overfitting.

Why might separability exist at all? We offer an empirically grounded (though not fully formal) explanation consistent with modern training paradigms. For verifiable tasks, reinforcement learning with outcome-based rewards creates repeated *contrast* between successful and failed internal trajectories. Such contrastive pressure encourages the model to represent "solution-consistent" versus "solution-inconsistent" intermediate states differently, making downstream distinctions more linearly accessible in deep layers (as we observe in §4, layer-wise analysis). Supervised fine-tuning, by comparison, predominantly imitates correct traces and provides much weaker negative feedback, which can leave "wrongness" underrepresented in the internal geometry. This hypothesis is consistent with our cross-model results showing RL-tuned models acting as stronger verifiers.

4 Experiments

To rigorously evaluate the effectiveness of our non-parametric, hidden-state-based verifier, we designed a series of experiments targeting both in-domain mathematical reasoning and out-of-distribution general reasoning tasks. Our evaluation is structured around two primary objectives: first, to assess the raw classification accuracy of our method in distinguishing correct from incorrect solutions, and second, to measure its ability to improve overall reasoning performance by reranking multiple candidate solutions.

4.1 Datasets and Model Configuration

Our methodology relies on an *experience set* to define geometric reference points for successful and failed reasoning. We construct the experience set from two standard mathematical benchmarks: AIME 1983–2023 (Veeraboina, 2023) and MATH (Hendrycks et al., 2021) (levels 3–5). For evaluation, we report in-domain results on AIME 2024 and AIME 2025, and out-of-domain generalization on GPQA (Rein et al., 2024), which tests graduate-level scientific reasoning beyond mathematics.

We evaluate three reasoning models spanning scales: **Nemotron-Research-Reasoning-Qwen-1.5B** (Liu et al., 2025), **Polaris-4B** (An et al., 2025), and **DeepSeek-R1-0528-Qwen3-8B** (Guo et al., 2025a). To study sensitivity to reasoning budget, we vary the maximum generation length (16k, 32k, 64k tokens) when applicable. Unless otherwise noted, we follow each model's recommended inference settings (e.g., temperature and system prompt) from its Hugging Face release.

We build the experience set separately for each reasoner model. For each problem in the AIME and MATH pools, we sample 32 solutions from that model and label each solution with a deterministic verifier when available (e.g., exact-answer checking for AIME/MATH). We then subsample a balanced set of 10,000 correct and 10,000 incorrect trajectories to compute the success and failure centroids for that model. For evaluation, we generate 64 candidate solutions per test problem.

4.2 Evaluation Setups and Baselines

We evaluate our method, which we refer to as **CLUE**, across two distinct experimental setups.

Table 1: Binary classification performance of different verifiers on solutions generated by Nemotron-1.5B and Polaris-4B. Our method (CLUE) is compared against an LLM-as-a-judge baseline. We report overall Accuracy, True Positive Rate (TPR), and True Negative Rate (TNR).

	Nemotron-1.5B Solutions			Polaris-4B Solutions		
Verifier Method	**Accuracy (%)**	**TPR (%)**	**TNR (%)**	**Accuracy (%)**	**TPR (%)**	**TNR (%)**
Test Set: AIME 2024						
CLUE (Ours)	**80.9**	72.9	87.4	**81.1**	89.5	51.3
GPT-4o (Answer Only)	58.6	57.2	59.7	80.1	84.3	63.1
GPT-4o (Full Trace)	47.5	45.8	48.2	64.3	70.9	50.2
Test Set: AIME 2025						
CLUE (Ours)	**85.2**	82.9	86.4	**77.7**	80.7	70.1
GPT-4o (Answer Only)	59.2	60.8	58.3	73.0	85.1	34.4
GPT-4o (Full Trace)	47.1	48.6	46.7	59.3	69.8	27.6

The first setup frames the task as a binary classification problem to directly measure the verifier's accuracy. For each of the 64 sampled solutions on the test sets, our CLUE method predicts a label of correct or incorrect based on whether the solution's activation delta is closer to the success or failure centroid. Labels are obtained from a rule-based verification signal when available (e.g., deterministic checking), and otherwise from reference-assisted verification (as described for WebInstruct in §4).

LLM-as-a-judge baseline. To ground comparison against a strong, widely used judge, we use **GPT-4o** (Hurst et al., 2024) in an LLM-as-a-judge capacity. We evaluate the judge in two settings to control for the information it can access: one where the full solution, including the entire `<think>` block, is provided, and another where only the final answer (post-`</think>`) is provided. Importantly, we use a *fixed, explicit scoring prompt* (reported verbatim in Appendix B) and deterministic decoding ($T{=}0$) to make the baseline as strong and reproducible as possible.

On baseline coverage. There is a large literature on learned and prompt-engineered judges (e.g., fine-tuned reward models, pairwise preference judges, and specialized LLM-judge prompting recipes). Incorporating every SOTA judge variant would require additional training and extensive prompt sweeps. Our intent here is more targeted: to test whether a frozen model's hidden states already contain a robust correctness signal that can be extracted by a low-capacity, non-parametric rule. We therefore include GPT-4o as a representative strong external judge and DeepConf as a representative confidence-based reranker, and position more exhaustive judge comparisons as future work.

Metrics. Because the positive/negative ratio can vary substantially with the underlying reasoner (especially when evaluating 64 samples per problem), we report Accuracy along with TPR/TNR to expose optimistic or pessimistic biases. We note that Matthews Correlation Coefficient (MCC) is also appropriate under class imbalance; reporting MCC requires the full confusion matrix counts, which we plan to release alongside code.

The second setup evaluates the practical impact of our method on improving final reasoning accuracy through reranking. Here, for each test problem, we use CLUE to rerank the 64 generated solutions. The ranking criterion is the Euclidean distance of a solution's activation delta to the success centroid, with smaller distances indicating higher quality. We report our performance using several metrics: **top@1**, the accuracy of the single best-ranked solution; and **top-maj@k**, the accuracy achieved by performing majority voting on the answers from the top-k ranked solutions, for $k \in \{4, 8, 16\}$. We compare these results against a suite of standard and state-of-the-art baselines. These include **mean@64**, which measures the average accuracy of a single randomly sampled solution; **majority@64**, the accuracy of standard majority voting over all 64 samples; **DeepConf@64** (Fu et al., 2025b), a recent and competitive method that uses model confidence scores for reranking; and **pass@64**, which represents the oracle upper bound, indicating whether at least one correct answer exists among the 64 samples.

4.3 Classification Performance

We first evaluate our method, CLUE, on its core capability: accurately classifying individual solutions as either correct or incorrect. Table 1 presents the

Table 2: Reasoning accuracy on AIME and GPQA test sets after reranking 64 candidate solutions. Results are presented as percentages (%). ‘mean@64‘ represents the average accuracy of a single sample, while ‘pass@64‘ is the oracle upper bound.

	Nemotron-1.5B			Polaris-4B			DeepSeek-8B		
Metric	**AIME 24**	**AIME 25**	**GPQA**	**AIME 24**	**AIME 25**	**GPQA**	**AIME 24**	**AIME 25**	**GPQA**
Baselines									
mean@64	45.0	35.0	41.9	79.2	75.4	55.2	87.1	75.8	54.86
majority@64	56.7	36.7	44.4	80.0	80.0	56.6	90.0	83.3	61.11
DeepConf@64	56.7	30.0	40.2	80.0	73.3	55.7	**93.3**	**86.7**	62.12
pass@64 (Oracle)	76.7	63.3	83.8	86.7	90.0	88.4	93.3	93.3	94.85
CLUE *Reranking (Ours)*									
top@1	66.7	40.0	46.5	**83.3**	76.7	52.5	90.0	83.3	56.57
top-maj@4	**70.0**	40.0	43.9	**83.3**	76.7	57.1	90.0	**86.7**	61.62
top-maj@8	**70.0**	40.0	**47.0**	80.0	80.0	58.1	**93.3**	**86.7**	61.11
top-maj@16	**70.0**	**43.3**	44.4	80.0	**83.3**	**59.6**	**93.3**	**86.7**	**62.63**

performance of our verifier compared to a strong LLM-as-a-judge baseline (GPT-4o) on solutions generated by both a smaller model (Nemotron-1.5B) and a more capable one (Polaris-4B). We report overall accuracy, as well as the True Positive Rate (TPR), which measures the ability to correctly identify successful solutions, and the True Negative Rate (TNR), which measures the ability to correctly identify failed solutions.

Beyond in-domain math classification, we also report an out-of-domain classification evaluation on the multi-domain WebInstruct-verified benchmark (Table 3), where correctness is grounded by reference answers. This directly addresses whether the same hidden-state signal transfers to heterogeneous, non-mathematical reasoning.

The results show that CLUE provides a consistent advantage over the LLM-as-a-judge baseline. A notable pattern is that the judge tends to be optimistic, often labeling incorrect solutions as correct; this is reflected by low True Negative Rates (e.g., 34.4% on Polaris-4B solutions for AIME 2025). This optimism also explains why the judge can appear stronger when the underlying reasoner is stronger: as the fraction of correct samples increases, high TPR contributes more to accuracy while the judge’s weakness on negatives matters less. In contrast, CLUE is more balanced across models, maintaining high TNR on weaker reasoners (filtering abundant failures) while retaining high TPR on stronger ones.

4.4 Reranking for Enhanced Reasoning Accuracy

Moving beyond binary classification, we evaluate CLUE as a reranking tool. By scoring and reordering 64 candidate solutions per problem, CLUE consistently improves over majority voting on both in-domain AIME and out-of-domain GPQA (Table 2). For instance, with Nemotron-1.5B on AIME 24, top-maj@16 reaches 70.0% versus 56.7% for majority@64. In several settings, top@1 also surpasses majority voting, indicating that distance to the success centroid is a useful selection signal.

The advantage extends to general reasoning: on GPQA, Polaris-4B reaches 59.6% with CLUE versus 56.6% with majority voting. Compared with the confidence-based baseline DeepConf, CLUE is more robust across model scales: while both methods perform strongly on DeepSeek-8B, DeepConf degrades substantially on weaker models (often below majority voting), whereas CLUE maintains gains, consistent with confidence miscalibration being more pronounced at smaller scales.

4.5 Generalization and the Influence of Training Paradigms

We next examine CLUE’s behavior across training paradigms and models. We hypothesize that the geometric separability of success and failure depends on training methodology, in particular the contrast between Supervised Fine-Tuning (SFT) and Reinforcement Learning (RL). We evaluate four models: two SFT/distillation-based (**Deepseek-7B** (Guo et al., 2025a), **Qwen3-4B** (Yang et al., 2025a)) and two RL-tuned (**Nemotron-1.5B**, **Polaris-4B**). In a cross-model setup, trajectories generated by one model are scored using the centroids and hidden states of another, enabling both self- and cross-verification tests.

As shown in Figure 3, SFT models struggle: their self-reranking (“top-maj@16”) barely

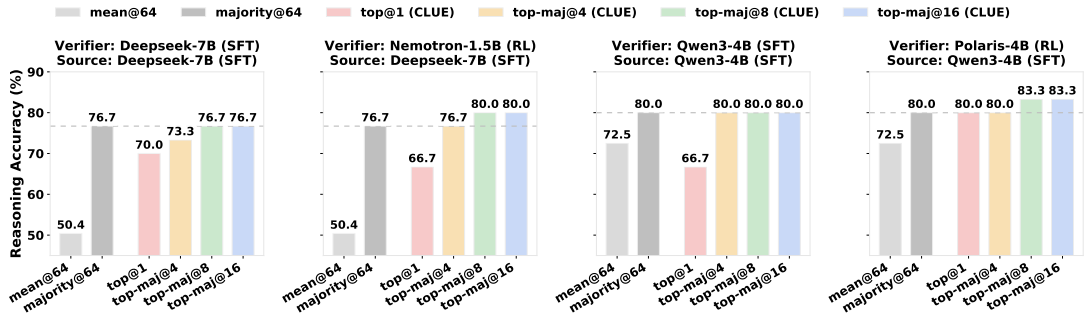

Figure 3: Cross-model reranking performance on AIME 24. RL-trained models (Nemotron-1.5B, Polaris-4B) are effective at self-verification and often act as stronger verifiers for trajectories generated by SFT-trained models (Deepseek-7B, Qwen3-4B).

matches or even lags the "majority@64" baseline, indicating weak internal separation of correctness. By contrast, RL models act as strong verifiers even across models: Nemotron-1.5B boosts Deepseek-7B's accuracy to 80.0% (vs 76.7% baseline), and Polaris-4B lifts Qwen3-4B's outputs to 83.3% (vs 80.0% self-rerank).

We attribute this gap to training signals. SFT predominantly imitates correct traces and provides limited direct feedback on failure modes. RL with rewards, in contrast, supplies repeated outcome-based contrast between success and failure, which can encourage a clearer internal separation.

4.6 Generalization to Diverse, Non-Mathematical Reasoning

To test CLUE's generalization beyond mathematics, we evaluate on the diverse **WebInstruct-verified** benchmark, spanning physics, law, finance, and the humanities. We build centroids from 5k training questions (with generated solutions) and evaluate on 1k test questions. Because deterministic checkers are unavailable, we obtain binary labels via *reference-assisted verification*: an evaluator (GPT-4o) is provided the reference answer to judge whether the candidate solution is correct. We use GPT-4o without reference access as the LLM-as-a-judge baseline. Since the same evaluator family is involved in both labeling and the judge baseline (albeit with different information access), these labels should be viewed as a pragmatic proxy rather than an independent ground-truth signal.

As shown in Table 3, CLUE outperforms GPT-4o across both 1.5B and 4B models. On the 1.5B model, CLUE reaches 60.4% accuracy versus GPT-4o's 54.0%. For the 4B model, the judge accuracy drops to 48.1%, while CLUE reaches 59.2%.

These results suggest that hidden-state geometry can provide a useful correctness signal even outside mathematics. Compared with surface-level textual judgments, which can be brittle in heterogeneous domains, CLUE leverages internal representations that appear more stable under distribution shift.

Table 3: Binary classification performance on the general-purpose WebInstruct-verified dataset. We compare CLUE's accuracy against a GPT-4o judge on solutions generated by 1.5B and 4B models. The centroids for CLUE were computed using the WebInstruct (Ma et al., 2025) training set.

	Reasoner Model	
Verifier Method	**Nemotron-1.5B**	**Polaris-4B**
CLUE (Ours)	**60.4%**	**59.2%**
GPT-4o	54.0%	48.1%

5 Conclusion

In this work, we investigate hidden states as a source of signal for correctness verification. We introduce CLUE, a training-free, experience-based verifier that summarizes each reasoning trace by an activation delta and performs nearest-centroid classification in hidden-state space. Across math, scientific QA, and a general-domain benchmark, CLUE improves verification and reranking compared with representative text-based and confidence-based baselines, with particularly strong gains on smaller or less-calibrated models. We further observe that RL-tuned reasoning models tend to exhibit clearer separation between successful and failed trajectories than SFT counterparts, suggesting that training paradigms can influence the geometry relevant to verification. An important direction for future work is to better characterize when and why such separability emerges, and how to design training objectives that improve verifier reliability in less strictly verifiable settings.

Limitations. CLUE relies on a supervision signal that can assign reliable success/failure labels to trajectories (e.g., exact match, deterministic checkers, or reference-grounded verification). In settings where factuality is inherently subjective or the notion of correctness is blurry, one may need preference labels or pairwise comparisons rather than binary correctness; we treat such settings as out of scope for the current study. Practically, CLUE also depends on an experience set: performance may vary with the quantity and diversity of labeled trajectories, and we discuss these design choices in §4. When deterministic checkers are unavailable and labels are produced by reference-assisted LLM evaluation, verification quality is naturally bounded by evaluator noise; we therefore treat such labels as a proxy rather than error-free ground truth.

References

Chenxin An, Zhihui Xie, Xiaonan Li, Lei Li, Jun Zhang, Shansan Gong, Ming Zhong, Jingjing Xu, Xipeng Qiu, Mingxuan Wang, and Lingpeng Kong. 2025. Polaris: A post-training recipe for scaling reinforcement learning on advanced reasoning models.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, and 1 others. 2022. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*.

Lukasz Bartoszcze, Sarthak Munshi, Bryan Sukidi, Jennifer Yen, Zejia Yang, David Williams-King, Linh Le, Kosi Asuzu, and Carsten Maple. 2025. Representation engineering for large-language models: Survey and research challenges. *Preprint*, arXiv:2502.17601.

Nora Belrose, Zach Furman, Logan Smith, Danny Halawi, Igor Ostrovsky, Lev McKinney, Stella Biderman, and Jacob Steinhardt. 2023. Eliciting latent predictions from transformers with the tuned lens. *arXiv preprint arXiv:2303.08112*.

Runjin Chen, Andy Arditi, Henry Sleight, Owain Evans, and Jack Lindsey. 2025. Persona vectors: Monitoring and controlling character traits in language models. *Preprint*, arXiv:2507.21509.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Tairan Fu, Javier Conde, Gonzalo Martínez, María Grandury, and Pedro Reviriego. 2025a. Multiple choice questions: Reasoning makes large language models (llms) more self-confident even when they are wrong. *arXiv preprint arXiv:2501.09775*.

Yichao Fu, Xuewei Wang, Yuandong Tian, and Jiawei Zhao. 2025b. Deep think with confidence. *arXiv preprint arXiv:2508.15260*.

Jiahui Geng, Fengyu Cai, Yuxia Wang, Heinz Koeppl, Preslav Nakov, and Iryna Gurevych. 2023. A survey of confidence estimation and calibration in large language models. *arXiv preprint arXiv:2311.08298*.

Amelia Glaese, Nat McAleese, Maja Trębacz, John Aslanides, Vlad Firoiu, Timo Ewalds, Maribeth Rauh, Laura Weidinger, Martin Chadwick, Phoebe Thacker, and 1 others. 2022. Improving alignment of dialogue agents via targeted human judgements. *arXiv preprint arXiv:2209.14375*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025a. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Jizhou Guo, Zhaomin Wu, Hanchen Yang, and Philip S. Yu. 2025b. Mining intrinsic rewards from llm hidden states for efficient best-of-n sampling. *Preprint*, arXiv:2505.12225.

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

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.

Arian Hosseini, Xingdi Yuan, Nikolay Malkin, Aaron Courville, Alessandro Sordoni, and Rishabh Agarwal. 2024. V-star: Training verifiers for self-taught reasoners. *arXiv preprint arXiv:2402.06457*.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, and 1 others. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, Scott Johnston, Sheer El-Showk, Andy Jones, Nelson Elhage, Tristan Hume, Anna Chen, Yuntao Bai, Sam Bowman, Stanislav Fort, and 17 others. 2022a. Language models (mostly) know what they know. *Preprint*, arXiv:2207.05221.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli

Tran-Johnson, and 1 others. 2022b. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*.

Tim Knappe, Ryan Li, Ayush Chauhan, Kaylee Chhua, Kevin Zhu, and Sean O'Brien. 2024. Semantic self-consistency: Enhancing language model reasoning via semantic weighting. *arXiv preprint arXiv:2410.07839*.

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

Zhen Lin, Shubhendu Trivedi, and Jimeng Sun. 2023. Generating with confidence: Uncertainty quantification for black-box large language models. *arXiv preprint arXiv:2305.19187*.

Mingjie Liu, Shizhe Diao, Ximing Lu, Jian Hu, Xin Dong, Yejin Choi, Jan Kautz, and Yi Dong. 2025. Prorl: Prolonged reinforcement learning expands reasoning boundaries in large language models. *arXiv preprint arXiv:2505.24864*.

Sheng Liu, Haotian Ye, Lei Xing, and James Zou. 2024. In-context vectors: Making in context learning more effective and controllable through latent space steering. *Preprint*, arXiv:2311.06668.

Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Meiqi Guo, Harsh Lara, Yunxuan Li, Lei Shu, Yun Zhu, Lei Meng, Jiao Sun, and Abhinav Rastogi. 2024. Improve mathematical reasoning in language models by automated process supervision. *Preprint*, arXiv:2406.06592.

Xueguang Ma, Qian Liu, Dongfu Jiang, Ge Zhang, Zejun Ma, and Wenhu Chen. 2025. General-reasoner: Advancing llm reasoning across all domains. *arXiv preprint arXiv:2505.14652*.

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

nostalgebraist. 2020. interpreting gpt: the logit lens. LessWrong post.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

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

Xuan Shen, Yizhou Wang, Xiangxi Shi, Yanzhi Wang, Pu Zhao, and Jiuxiang Gu. 2025. Efficient reasoning with hidden thinking. *arXiv preprint arXiv:2501.19201*.

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

Lorenzo Tomaz, Judd Rosenblatt, Thomas Berry Jones, and Diogo Schwerz de Lucena. 2025. Momentum point-perplexity mechanics in large language models. *arXiv preprint arXiv:2508.08492*.

Hemish Veeraboina. 2023. Aime problem set 1983-2024.

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

Miao Xiong, Andrea Santilli, Michael Kirchhof, Adam Golinski, and Sinead Williamson. 2024. Efficient and effective uncertainty quantification for llms. In *Neurips Safe Generative AI Workshop 2024*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025a. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

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

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822.

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

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, and 1 others. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623.

A Algorithmic Details

For completeness, we provide pseudocode for the two phases of CLUE: the one-time centroid aggregation (Algorithm 1) and the inference-time verification (Algorithm 2). Both phases operate on *activation-delta matrices* (§3.2) and use the *layer-averaged Euclidean distance* in Eq. (4). The underlying LLM remains frozen throughout.

Algorithm 1 constructs the reference centroids from a labeled set of trajectories. We first partition the dataset by ground-truth labels. For each trajectory, we extract the hidden-state matrices at the boundaries of the explicit reasoning block (`<think>` and `</think>`) and compute the activation delta $\Delta\mathbf{h}_i \in \mathbb{R}^{L\times D}$ via Eq. (1). We then compute the element-wise mean within each class to obtain the success and failure *centroid matrices* $\mathbf{V}_{\text{succ}}$ and $\mathbf{V}_{\text{fail}}$ (Eq. (3)), which serve as geometric references during inference.

Algorithm 1 Constructing CLUE Centroids (Learning Phase)

Require: Labeled dataset $\mathcal{D} = \{(T_i, y_i)\}_{i=1}^{N}$
Initialize empty lists $\mathcal{H}_{\text{succ}}$, $\mathcal{H}_{\text{fail}}$
Define index sets $\mathcal{I}_{\text{succ}} = \{\, i \mid y_i = 1 \,\}, \mathcal{I}_{\text{fail}} = \{\, i \mid y_i = 0 \,\}$
for each $i \in \mathcal{I}_{\text{succ}}$ **do**
    Extract $\mathbf{h}_{\text{start},i} \in \mathbb{R}^{L\times D}$ and $\mathbf{h}_{\text{end},i} \in \mathbb{R}^{L\times D}$
    Compute $\Delta\mathbf{h}_i \leftarrow \mathbf{h}_{\text{end},i} - \mathbf{h}_{\text{start},i}$ (Eq. 1)
    Append $\Delta\mathbf{h}_i$ to $\mathcal{H}_{\text{succ}}$
end for
for each $i \in \mathcal{I}_{\text{fail}}$ **do**
    Extract $\mathbf{h}_{\text{start},i}$ and $\mathbf{h}_{\text{end},i}$
    Compute $\Delta\mathbf{h}_i \leftarrow \mathbf{h}_{\text{end},i} - \mathbf{h}_{\text{start},i}$
    Append $\Delta\mathbf{h}_i$ to $\mathcal{H}_{\text{fail}}$
end for
$\mathbf{V}_{\text{succ}} \leftarrow \text{mean}(\mathcal{H}_{\text{succ}})$ (Eq. 3)
$\mathbf{V}_{\text{fail}} \leftarrow \text{mean}(\mathcal{H}_{\text{fail}})$ (Eq. 3)
return $\mathbf{V}_{\text{succ}}$, $\mathbf{V}_{\text{fail}}$

B LLM-as-a-Judge Prompt

To make the GPT-4o baseline reproducible and as strong as possible without prompt sweeping, we use a fixed, explicit rubric-style prompt and deterministic decoding (T=0). We provide it verbatim below.

> You are a strict verifier for reasoning problems. You will be given a problem and a candidate solution. Your job is to decide whether the final answer is correct.
>
> Instructions: 1) Focus on mathematical/logical correctness, not writing style. 2) If the final answer is missing, ambiguous, or does not follow from the reasoning, mark it INCORRECT. 3) If any step contains a clear logical or arithmetic error that invalidates the final answer, mark it INCORRECT. 4) If the reasoning is incomplete but the final answer is clearly correct and consistent with the problem, you may mark it CORRECT. 5) Output ONLY one token: CORRECT or INCORRECT.
>
> Problem: {PROBLEM}
>
> Candidate solution: {SOLUTION}

For the "Answer Only" setting, we set `{SOLUTION}` to the text after `</think>`. For the "Full Trace" setting, we include the entire model output (including the `<think>` block).

C Experience Set Construction and Hyperparameters

Our verifier depends on a labeled experience set and a small number of practical design choices. We briefly justify them here.

- **Experience set size and balance.** We use a balanced set (equal numbers of successes and failures) to avoid trivial centroid shifts driven by class frequency. Increasing the size primarily reduces centroid estimation noise; diminishing returns are expected once the centroids stabilize under additional samples.
- **Rollouts per prompt.** More rollouts increase diversity of trajectories (including near-miss failures), which can sharpen the boundary between success/failure prototypes. However, they also increase compute and may introduce many redundant trajectories; our setting is chosen as a practical trade-off.
- **Layer aggregation.** Our layer-wise analysis shows stronger separability in deeper layers, but we aggregate across layers to reduce sensitivity to idiosyncrasies of any single layer and to make the verifier more stable across architectures and training paradigms. A thorough ablation over layer subsets is left for future work.

Algorithm 2 describes the inference procedure. Given a new trajectory T_{new}, we compute its activation delta $\Delta\mathbf{h}_{\text{new}}$ as in Eq. (1), measure its distances to the two centroid matrices using Eq. (4), and decide by nearest centroid.

Algorithm 2 Verification with CLUE (Inference Phase)

Require: New trajectory T_{new}; centroids $\mathbf{V}_{\text{succ}}$, $\mathbf{V}_{\text{fail}}$

Extract $\mathbf{h}_{\text{start,new}}$ and $\mathbf{h}_{\text{end,new}}$ from T_{new}
Compute $\Delta\mathbf{h}_{\text{new}} \leftarrow \mathbf{h}_{\text{end,new}} - \mathbf{h}_{\text{start,new}}$ (Eq. 1)
$d_{\text{succ}} \leftarrow d(\Delta\mathbf{h}_{\text{new}}, \mathbf{V}_{\text{succ}})$ (Eq. 4)
$d_{\text{fail}} \leftarrow d(\Delta\mathbf{h}_{\text{new}}, \mathbf{V}_{\text{fail}})$ (Eq. 4)
if $d_{\text{succ}} < d_{\text{fail}}$ **then**
    return 1 ▷ classified as correct
else
    return 0 ▷ classified as incorrect
end if

C.1 Layer-wise Separability Analysis

Next, we analyze the layer-wise structure of activation-delta matrices to visualize and quantify how class separability emerges from shallow to deep layers.

Visualization. We project layer-specific activation deltas onto two principal components via PCA. For a trajectory i and layer ℓ, let $\Delta\mathbf{h}_i^{(\ell)} \in \mathbb{R}^D$ denote the ℓ-th row of $\Delta\mathbf{h}_i \in \mathbb{R}^{L\times D}$. We select representative shallow, middle, and final layers, apply PCA to $\{\Delta\mathbf{h}_i^{(\ell)}\}$, and plot the resulting 2D projections for successes and failures. As shown in the first three columns of each row in Figure 4, the classes are largely overlapping in shallow layers, begin to separate in middle layers, and form compact, well-defined clusters in the final layers.

Quantification. Let $\mathcal{I}_{\text{succ}}$ and $\mathcal{I}_{\text{fail}}$ be the index sets defined in §3.3. For each layer $\ell \in \{1, \ldots, L\}$, we compute layer-wise centroid by averaging the corresponding rows of the activation-delta matrices:

$$\mathbf{V}_{\text{succ}}^{(\ell)} = \frac{1}{|\mathcal{I}_{\text{succ}}|} \sum_{i\in\mathcal{I}_{\text{succ}}} \Delta\mathbf{h}_i^{(\ell)} \in \mathbb{R}^D, \tag{6}$$

$$\mathbf{V}_{\text{fail}}^{(\ell)} = \frac{1}{|\mathcal{I}_{\text{fail}}|} \sum_{i\in\mathcal{I}_{\text{fail}}} \Delta\mathbf{h}_i^{(\ell)} \in \mathbb{R}^D. \tag{7}$$

We then measure the Euclidean distance between the two centroids at layer ℓ:

$$d^{(\ell)} = \left\|\mathbf{V}_{\text{succ}}^{(\ell)} - \mathbf{V}_{\text{fail}}^{(\ell)}\right\|_2. \tag{8}$$

The rightmost panels of Figure 4 plot $d^{(\ell)}$ across layers. We observe a consistent upward trend, with the distance typically peaking in the final layers, aligning with the PCA visualizations and indicating that deeper representations encode more explicit and separable correctness signals.

D Additional Ablation Studies

To validate the specific design choices of our CLUE methodology, we conducted a series of ablation studies. Our goal was to isolate the contributions of two key components: 1) the use of hidden states from all layers of the model, and 2) the computation of an activation *delta* ($\mathbf{h}_{\text{end}} - \mathbf{h}_{\text{start}}$) rather than using an absolute state vector. We evaluated three alternative configurations against our full method in Figure 5:

- **First Layer Only**: Using only the hidden states from the first transformer layer.
- **Last Layer Only**: Using only the hidden states from the final transformer layer.
- **Final State Only**: Using only the absolute hidden state vector at the end of the reasoning block ($\mathbf{h}_{\text{end}}$), without subtracting the baseline state.

The results of our ablation study confirm that each component of the CLUE verifier contributes to its overall effectiveness. The most significant finding is the critical role of network depth. Using only the **First Layer Only** leads to a dramatic performance collapse across all settings, indicating that the shallow, near-embedding layers of the model do not contain sufficiently abstract or separable representations to distinguish correct from incorrect reasoning. Conversely, the **Last Layer Only** variant performs remarkably well, achieving accuracy that is only marginally lower than our full method and even matching it in some cases. This demonstrates that the deepest layers of the model are the primary locus of the high-level reasoning signals we are leveraging. Finally, the **Final State Only** experiment highlights the benefit of our delta computation. While still a strong performer, removing the subtraction of the baseline state results in a noticeable performance degradation compared to the full CLUE approach.

In summary, these ablations validate our methodology: the discriminative signal is strongest in deep layers, but leveraging the full stack of layers and calculating the activation delta are both important for achieving optimal performance.

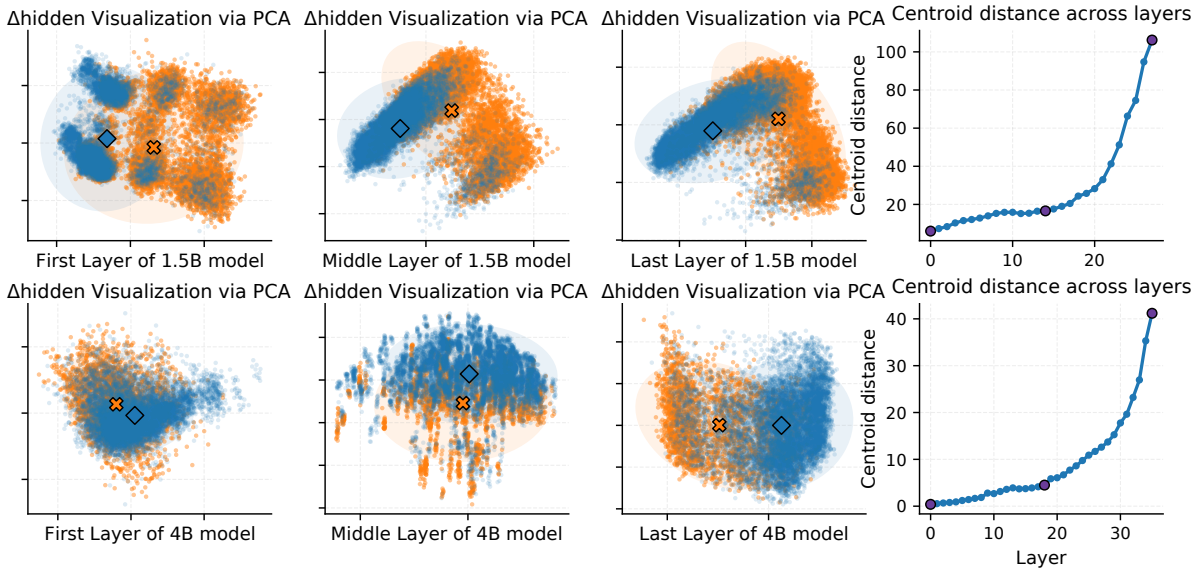

Figure 4: Layer-wise separability. Each row shows PCA projections from a shallow, a middle, and the final layer, plus a curve of the centroid distance $d^{(\ell)}$ across all layers. The centroid-distance curve increases with ℓ, indicating stronger correctness signals at deeper layers.

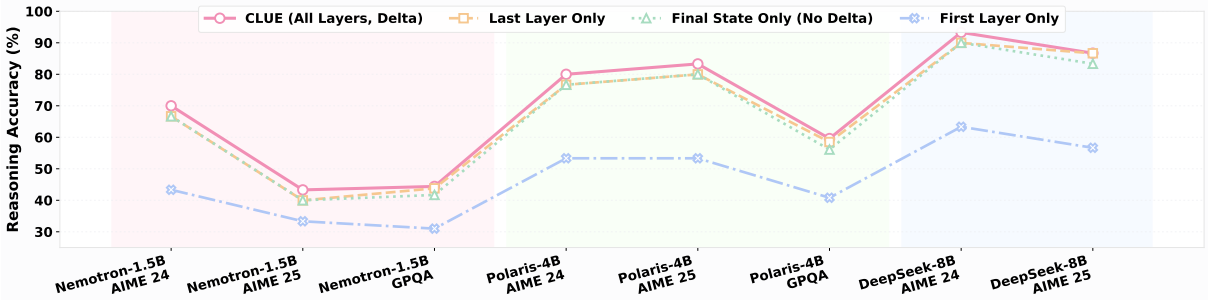

Figure 5: Ablation study results showing the ‘top-maj@16‘ reasoning accuracy across different model and dataset configurations.