RETHINKING LLMS AS VERIFIERS: WHEN VERIFICATION IS HARDER THAN SOLVING

Anonymous authors
Paper under double-blind review

ABSTRACT

Large Language Models (LLMs) are increasingly used as evaluators of model outputs, a paradigm commonly referred to as LLM-as-a-judge. A natural assumption underlying this paradigm is that verification should be easier than solving: given a candidate solution, a model should reliably determine its correctness. In this work, we empirically test this assumption. Across multiple benchmarks and model families, we find that LLMs are often less accurate at verification than at solving the same tasks. This gap persists across domains, including multiple-choice reasoning, program synthesis, and multi-step problem solving. To understand this failure, we study verification along three axes. First, we identify *epistemic bias*, where models are more reliable at accepting correct solutions than rejecting incorrect ones. Second, we show *perturbation insensitivity*, where models fail to detect localized errors in near-correct solutions. Third, we demonstrate that verification accuracy improves with *rubric conditioning*, highlighting the role of explicit evaluation criteria. Our results show that LLM-based evaluation is not a straightforward proxy for correctness. Instead, it exhibits systematic failure modes that must be accounted for when using LLMs as evaluators in post-training and benchmarking pipelines.

1 INTRODUCTION

Large Language Models (LLMs) have rapidly evolved from generative systems to general-purpose reasoning agents, capable of performing complex tasks across domains (Brown et al., 2020; Chowdhery et al., 2023). A key driver of this progress has been post-training, where models are aligned and improved using techniques such as supervised fine-tuning and preference optimization (Ouyang et al., 2022; Rafailov et al., 2023; Guo et al., 2025). These methods increasingly rely on implicit or learned reward signals, rather than explicit ground-truth supervision, shifting the burden of evaluation from humans to models themselves (Bai et al., 2022; Lee et al., 2024).

This shift has led to the emergence of the *LLM-as-a-judge* paradigm (Li et al., 2024a), where models are used as autonomous evaluators of outputs produced by other models – or even themselves. Such evaluators are now central to modern pipelines, including reinforcement learning from AI feedback, self-improvement loops, and hallucination detection (Li et al., 2025; Huang et al., 2025). In these settings, LLMs are tasked not only with generating solutions but also with assessing their correctness, effectively acting as verifiers in the absence of external supervision.

A common intuition underlying this paradigm is that *verification should be easier than solving*. This view is loosely motivated by complexity-theoretic reasoning: given access to a solver, one can construct a verifier by independently recomputing the solution and checking consistency. By analogy, when the same model family is used for both generation and evaluation, one might expect verification to be at least as reliable as solving.

Solvers vs. Verifiers. We compare model performance when used as a solver against its performance when verifying candidate solutions. Figure 1 plots this comparison across *MMLU-Pro* (Wang et al., 2024) subjects, where the diagonal ($y = x$) denotes parity between solving and verification accuracy. A substantial fraction of subjects lies below the diagonal, indicating that verification accuracy is lower than solving accuracy. This pattern holds across subjects, suggesting that the drop

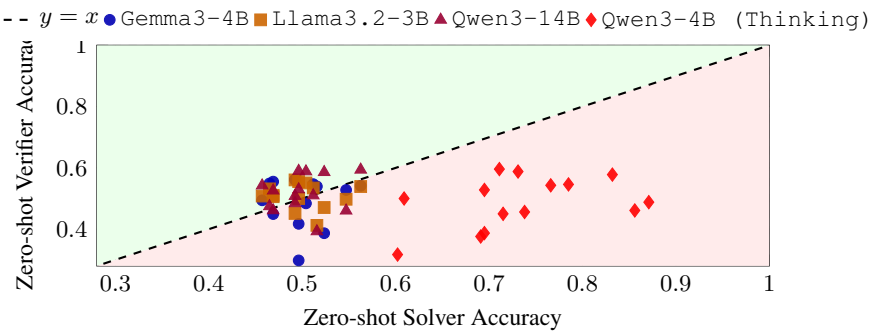

Figure 1: Verifier vs. solver accuracy across *MMLU-Pro* subjects. Each point represents average zero-shot accuracy of solver and verifier on a particular subject of *MMLU-Pro* for a particular model. The diagonal ($y = x$) separates regimes where verification matches solving. A substantial fraction of points lie below the diagonal (red region), indicating that the verifier under performs the solver, contradicting the intuition that verification is easier.

in verification accuracy is not tied to any particular model or task but is inherent to the LLM-based verification setup.

Failure Modes of Verification. Aggregate accuracy alone does not explain this gap. Instead, verification errors arise from systematic properties of model-based evaluation. We identify three distinct failure modes: (i) *epistemic bias*, where models preferentially accept plausible but incorrect solutions; (ii) *perturbation insensitivity*, where models fail to detect localized errors in near-correct outputs; and (iii) *rubric conditioning*, where performance depends critically on the structure of the evaluation criteria.

Prior work has examined the reliability of LLM-based evaluators, identifying issues such as position bias, prompt sensitivity, and disagreement with human judgments (Shi et al., 2025; Wei et al., 2025; Huang et al., 2025; Fu & Liu, 2025). However, this line of work primarily focuses on comparative evaluation, such as ranking or preference modeling. Concurrent work like Lu et al. (2025) attempts to study the challenges of using LLM-as-a-judge by observing solver-verifier variance, with changing solver and verifier model-sizes and model-architectures. We study *verification* as a binary decision problem with ground-truth correctness, and provide a systematic analysis of its failure modes.

Contributions. We introduce a three-axis evaluation framework that isolates distinct failure modes of LLM-as-a-Judge. Along *Axis I (Epistemic Bias)*, we measure the gap between true positive and true negative rates across 12 model–dataset configurations and find a pervasive accept bias: normalized bias Δ_{norm} reaches $+0.96$ for `Qwen3-4B (Thinking)` on *HumanEval* (TPR = 0.99, TNR = 0.35) and $+0.78$ for `Gemma3-4B` on *MMLU-Pro* (TPR = 0.73, TNR = 0.32), with only two configurations exhibiting a mild reject bias (`Llama3.2-3B` on *CHAMP* at -0.12 and `Qwen3-14B` on *MMLU-Pro* at -0.08). Along *Axis II (Perturbation Sensitivity)*, we show that rejection accuracy drops sharply for near-correct solutions: at 1 edit, `Gemma3-4B` detects only 59.2% of functionally broken code compared to 87.8% at 20 edits, and even the best-performing model (`Qwen3-4B (Thinking)`) falls from 100% to 90.2%. Along *Axis III (Rubric Conditioning)*, we demonstrate that structured evaluation criteria recover verification accuracy, with `Llama3.2-3B` improving from 20.4% (k=0) to 92.8% (k=6) and `Gemma3-4B` from 45.7% to 74.6%, while `Qwen3-4B (Thinking)`—already at 85.5%—plateaus near 84–86%, suggesting rubrics primarily compensate for weaker internal reasoning.

2 Related Work

LLM-as-a-Judge. Recent work has demonstrated that LLMs can act as scalable evaluators for model outputs, enabling new forms of training and evaluation pipelines. In particular, LLM-based

judging has been used to automate large-scale benchmarking and reduce reliance on human annotation (Li et al., 2024a).

LLM-based judges have been used to perform rubric-based scoring and preference comparisons for open-ended generation tasks, enabling scalable evaluation without human annotators (Liu et al., 2023a; Kim et al., 2023; Li et al., 2024b). Beyond evaluation, several methods rely on LLM-generated judgments as supervision signals during training. For example, Constitutional AI and related approaches replace human feedback with AI-generated critiques or preferences (Bai et al., 2022). Large-scale feedback datasets such as UltraFeedback leverage strong models to produce synthetic preference annotations (Cui et al., 2024). Similarly, critique-based frameworks train or deploy LLM critics to assess and refine model outputs, as in Shepherd (Wang et al., 2023a), Self-Refine (Madaan et al., 2023), and Reflexion (Shinn et al., 2023). More recently, Direct Judgement Preference Optimization use model-generated judgments directly to guide policy learning (Wang et al., 2025). Together, these approaches highlight the growing reliance on LLM-based judges as scalable sources of evaluation and supervision.

Limitations & Biases in LLM-based Judges. Recent work has begun to examine the reliability and limitations of LLM-based evaluators. Li et al. (2025) provide a comprehensive study of LLM-as-a-judge systems, identifying systematic issues such as positional bias, prompt sensitivity, and inconsistencies in evaluation across tasks. Similarly, Sun et al. (2024) analyze the reliability of critique-based evaluation and show that LLM-generated critiques can be incomplete or misleading, limiting their usefulness as reliable feedback signals. Complementing these analyses, Lin et al. (2024) introduce CriticBench, a benchmark designed to measure LLM capabilities in critique and correction reasoning across multiple domains, highlighting gaps between generation and critique abilities. In a related direction, Feng et al. (2025) study the reliability of LLM-based evaluators through consistency-based metrics that measure stability and transitivity of model judgments, showing that LLM judges frequently exhibit inconsistent preferences.

Beyond evaluation-specific studies, recent work in reasoning benchmarks and verification settings further highlights limitations in LLMs' ability to reliably assess correctness. The CHAMP benchmark shows that models can arrive at correct answers through flawed reasoning and often fail to verify such solutions (Mao et al., 2024). Similarly, studies on NP-hard problem solving demonstrate that LLM performance is highly sensitive to problem presentation, suggesting brittle reasoning that may undermine reliable evaluation (Duchnowski et al., 2025). Work on fine-tuned judge models further shows limited generalization, with such models behaving as task-specific classifiers that fail to match stronger models across domains (Huang et al., 2025). Closest to our setting, Lu et al. (2025) study LLMs as solution verifiers and analyze solver-verifier interactions, showing that verification performance varies significantly across tasks and model families.

While these works primarily document empirical inconsistencies, brittleness, and benchmarking gaps, our work investigates a more fundamental question: whether verification itself is easier than generation for LLMs, and how biases in verification, such as optimism toward plausible but incorrect answers, affect the reliability of LLM-based judging.

3 METHODOLOGY

Modern language models are parameterized conditional distributions $p_\theta(y \mid x)$ trained on large-scale corpora and evaluated across a broad range of downstream tasks. A task is defined by an input space $\mathcal{X}$, an output space $\mathcal{Y}$, and a ground-truth mapping $f : \mathcal{X} \to \mathcal{Y}$. Given $x \sim \mathcal{D}$, a model generates $\hat{y} \sim p_\theta(\cdot \mid x)$, which is evaluated against $f(x)$ using task-dependent metrics.

While exact-match or executable metrics are available in restricted settings (e.g., multiple-choice or program synthesis with test cases), many tasks (particularly open-ended generation) lack reliable automatic evaluation. A common approach is to employ a second model to assess the correctness or quality of $\hat{y}$, effectively replacing explicit metrics with learned evaluation functions .

We formalize this as a two-stage pipeline:

$$\hat{y} \sim p_\theta^{\text{solve}}(\cdot \mid x), \quad \hat{z} \sim p_\phi^{\text{verify}}(\cdot \mid x, \hat{y}),$$

where p_θ^{solve} generates candidate solutions and p_ϕ^{verify} produces an evaluation signal. In the simplest case, $\hat{z} \in \{0, 1\}$ represents a binary correctness judgment.

This abstraction captures a range of evaluation paradigms, including reference-based scoring, preference modeling, and learned reward functions used in alignment pipelines . Crucially, the evaluation model p_ϕ^{verify} is itself a pre-trained distribution not explicitly optimized for verification, and therefore introduces an additional layer of approximation beyond the task mapping f.

We further consider *rubric-based evaluation* (Zheng et al., 2023; Bai et al., 2022), where the verifier is provided with structured criteria to guide its judgment. Formally, a rubric is a collection of criteria

$$\mathcal{R} = \{g_1, g_2, \ldots, g_k\},$$

where each g_i specifies a property that a correct solution should satisfy. For example, in a reasoning task, g_1 may require correctness of intermediate steps, while g_2 may enforce consistency with the final answer. The evaluation model is then conditioned on $\mathcal{R}$:

$$\hat{z} \sim p_\phi^{\text{verify}}(\cdot \mid x, \hat{y}, \mathcal{R}),$$

allowing verification to be decomposed into finer-grained checks.

The reliability of this pipeline depends not only on the quality of generated solutions but also on the inductive biases and failure modes of p_ϕ^{verify}. In this work, we study the verification process induced by p_ϕ^{verify} along three axes: (i) epistemic bias in distinguishing correct and incorrect solutions, (ii) sensitivity to localized perturbations of otherwise correct outputs, and (iii) the effect of structured evaluation criteria (rubrics) on verification accuracy.

3.1 Axis I: Epistemic Bias

We examine whether the evaluation model p_ϕ^{verify} exhibits asymmetric behavior when judging correct and incorrect candidate solutions. Ideally, verification should be symmetric: correct candidates are accepted and incorrect candidates are rejected with comparable reliability. However, p_ϕ^{verify} is a *pre-trained* model not optimized for verification, and its judgments may depend on properties of $\hat{y}$ beyond correctness, such as plausibility or alignment with common reasoning patterns (Wang et al., 2023b; Lightman et al., 2024).

We instantiate verification as a reduction of standard evaluation benchmarks with known ground-truth answers. For each input x with ground-truth $f(x)$, we form candidate solutions $\hat{y}$ which may be either correct ($\hat{y} = f(x)$) or incorrect ($\hat{y} \neq f(x)$). The evaluation model predicts a binary label:

$$\hat{z} \in \{0, 1\}, \quad \hat{z} = 1 \text{ iff } \hat{y} = f(x),$$

based on $(x, \hat{y})$.

We characterize verification performance using standard confusion statistics. Let TP and TN denote the rates at which correct and incorrect candidates are correctly classified, and FP and FN denote the rates at which incorrect candidates are accepted and correct candidates are rejected, respectively.

We define the *epistemic bias gap*[1] as

$$\Delta_{\text{bias}} = \text{TPR} - \text{TNR}.$$

A non-zero Δ_{bias} indicates asymmetric verification behavior. In particular, $\Delta_{\text{bias}} > 0$ implies that the evaluator is more reliable at confirming correct candidates than rejecting incorrect ones, reflecting an *acceptance bias* toward plausible but incorrect outputs. However, Δ_{bias} alone does not account for the model's overall competence. For instance, a high bias may arise simply due to uniformly strong or weak performance.

[1]We refer to this asymmetric acceptance rate (Δ_{bias}) as an 'epistemic bias' because it reflects a default to acceptance when the model lacks the necessary knowledge boundaries to confidently reject, rather than a mere calibration error.

Balanced Accuracy and Normalized Bias. In verification settings, performance must account for both the ability to accept correct solutions and reject incorrect ones. Let TPR denote the true positive rate (accepting correct solutions) and TNR denote the true negative rate (rejecting incorrect solutions). We measure overall verification capability using *balanced accuracy*:

$$\text{BalAcc} = \frac{\text{TPR} + \text{TNR}}{2}.$$

Unlike standard accuracy, balanced accuracy assigns equal importance to both classes and is invariant to class imbalance, making it appropriate for evaluating verification behavior.

To normalize for overall capability, we define the *normalized bias*:

$$\Delta_{\text{norm}} = \frac{\Delta_{\text{bias}}}{\text{TPR} + \text{TNR}} = \frac{\text{TPR} - \text{TNR}}{\text{TPR} + \text{TNR}}.$$

This metric captures the *relative asymmetry* in verification behavior, independent of absolute performance. Intuitively, Δ_{norm} measures how skewed a verifier is toward accepting solutions versus rejecting them, normalized by its overall balanced accuracy. High positive values indicate a systematic tendency to accept candidate solutions (even at the cost of failing to reject incorrect ones), while negative values indicate the opposite.

Thus, Δ_{norm} isolates epistemic bias from raw verification ability, enabling meaningful comparisons across models and datasets with differing accuracy levels.

This axis isolates a key failure mode: whether verification operates as semantic correctness checking or as plausibility filtering. By evaluating performance conditioned on ground-truth correctness, we directly quantify the extent to which p_ϕ^{verify} conflates these objectives.

3.2 Axis II: Perturbation Sensitivity

We study the sensitivity of the evaluation model p_ϕ^{verify} to localized errors in candidate solutions. Language models are known to produce outputs that are highly plausible yet incorrect, often differing from correct solutions in only small, localized ways (Wang et al., 2023b; Lightman et al., 2024). A reliable verifier should detect such errors even when the overall structure of the solution appears valid.

We formalize this by constructing perturbed variants of ground-truth solutions. For each input x with solution $y = f(x)$, we generate candidates $\hat{y}^{(k)}$ by applying k localized edits. These edits are introduced via deterministic transformations (e.g., token-level corruptions, datatype changes, or numerical modifications for coding tasks) that induce incorrectness while preserving the global structure and surface plausibility of the solution. By construction, $\hat{y}^{(k)} \neq f(x)$ for all $k \geq 1$.

Given $(x, \hat{y}^{(k)})$, the evaluation model produces a binary judgment $\hat{z}^{(k)} \in \{0, 1\}$, where $\hat{z}^{(k)} = 1$ denotes acceptance. We measure the verifier's ability to reject incorrect candidates via the hit rate:

$$H^{(k)} = \mathbb{E}\left[\mathbb{I}(\hat{z}^{(k)} = 0)\right].$$

The parameter k controls the proximity of $\hat{y}^{(k)}$ to the ground-truth solution. Small values of k correspond to *near-miss* candidates that remain structurally similar to correct solutions but contain localized errors, while larger values induce progressively larger deviations.

This setup isolates whether p_ϕ^{verify} performs fine-grained validation or relies on coarse structural cues. In particular, a decrease in $H^{(k)}$ for small k indicates that the verifier fails to detect subtle errors and instead accepts globally plausible candidates. Variation in $H^{(k)}$ directly reflects sensitivity to localized perturbations, independent of task difficulty or label ambiguity.

3.3 Axis III: Rubric Conditioning

We study whether conditioning the evaluation model p_ϕ^{verify} on structured criteria improves verification accuracy.

Given an input x, candidate solution $\hat{y}$, and rubric $\mathcal{R} = \{g_1, \ldots, g_n\}$ as defined in §3.1, we condition the verifier on subsets of criteria:

$$\hat{z}_k \sim p_\phi^{\text{verify}}(\cdot \mid x, \hat{y}, \mathcal{R}_k), \quad \mathcal{R}_k = \{g_1, \ldots, g_k\},$$

where $\hat{z}_k \in \{0, 1\}$ denotes the binary correctness judgment and k controls the level of specification.

We evaluate verification accuracy as a function of k:

$$A(k) = \mathbb{E}\left[\mathbb{I}(\hat{z}_k = \mathbb{I}(\hat{y} = f(x)))\right].$$

Increasing k corresponds to progressively refining the evaluation objective via more explicit criteria. Improvements in $A(k)$ indicate that rubric conditioning helps the verifier perform more accurate and consistent correctness judgments by decomposing the task into finer-grained checks.

Conversely, limited or diminishing gains as k increases suggest that verification errors are not solely due to underspecified evaluation objectives, but reflect intrinsic limitations of p_ϕ^{verify}. This axis, therefore, isolates the extent to which verification performance can be improved through better specification of evaluation criteria.

4 Setup & Experiments

4.1 Setup

Models. We evaluate a set of open-weight LMs: `Qwen3-14B` & `Qwen3-4B (Thinking)` (Yang et al., 2025), `Llama3.2-3B` (Meta AI, 2024), and `Gemma3-4B` (Google, 2025). Each model is used in two roles: (i) as a solver $p_\theta^{\text{solve}}(\cdot \mid x)$ to produce candidate solutions, and (ii) as a verifier $p_\phi^{\text{verify}}(\cdot \mid x, \hat{y})$ to perform verification. Unless otherwise specified, we use identical prompting and decoding settings across tasks and treat the verification output as a binary judgment $\hat{z} \in \{0, 1\}$.

Datasets. We evaluate on four benchmarks with ground-truth answers:

1. *MMLU-Pro* Wang et al. (2024): broad, subject-wise multiple-choice reasoning
2. *MedMCQA* (Wei et al., 2025): medical-domain multiple-choice QA
3. *HumanEval* (Chen et al., 2021): code generation with unit-test evaluation
4. *CHAMP* (Mao et al., 2024): multi-step reasoning with annotated hints

These datasets span multiple-choice QA, domain-specific reasoning, program synthesis, and multi-step problem solving (see §A for details). For each dataset, we cast the task as verification by pairing inputs x with candidate solutions $\hat{y}$ and requiring the model to predict whether $\hat{y}$ is correct. We construct a balanced verification set with a 50%–50% split of correct and incorrect candidates. Incorrect candidates are generated either by sampling alternative options (e.g., other choices in *MMLU-Pro*) or by perturbing correct solutions (e.g., modifying the reference code in *HumanEval*).

Axes. We instantiate the three evaluation axes described in §3 on the models and datasets mentioned above. For **Axis I**, we use the native tasks in *MMLU-Pro*, *MedMCQA*, *HumanEval*, and *CHAMP*, constructing candidate solutions $\hat{y}$ that are either correct or incorrect, and measure conditional hit rates over these instances. For **Axis II**, we focus on program synthesis and structured reasoning tasks (*HumanEval*, *CHAMP*), where we generate perturbed variants of ground-truth solutions via controlled edits, and evaluate rejection performance as a function of perturbation level. Details of each edit-type can be found in Table 1. For **Axis III**, we use *CHAMP*, which provides solution decompositions that enable the construction of rubric sets $\mathcal{R}$ from problem-specific hints. We then progressively condition the evaluator on increasing subsets $\mathcal{R}_k$ and measure verification accuracy as a function of k.

4.2 Results

Edit Type	Description	Example	Failure Mode
Character / Spelling	Replace characters or identifiers with visually similar variants	`count` → `c0unt`	Syntactic errors, unintended variable mismatch
Number Change	Substitute numeric constants with different values	`n = 10` → `n = 12`	Logical errors (incorrect computation)
Data Type Change	Modify variable or function data types	`int` → `float`	Type errors, runtime failures
Expression Change	Alter operators or expressions while preserving structure	`a + b` → `a - b`, `<` → `>=`	Logical errors (semantic change)

Table 1: Types of perturbations introduced in the Perturbation Sensitivity experiment setting. Each edit operates at a different abstraction level and induces distinct failure modes. In the experiment, an edit-type is randomly selected and applied to the code.

Dataset	Model	TPR	TNR	Δ_{bias}	Δ_{norm}
CHAMP	`Qwen3-4B (Thinking)`	0.86	0.54	+0.33	0.47
HumanEval	`Qwen3-4B (Thinking)`	0.99	0.35	+0.64	0.96
MMLU-Pro	`Qwen3-4B (Thinking)`	0.50	0.48	+0.02	0.04
CHAMP	`Llama3.2-3B`	0.55	0.61	-0.07	-0.12
HumanEval	`Llama3.2-3B`	0.63	0.54	+0.09	0.15
MMLU-Pro	`Llama3.2-3B`	0.67	0.41	+0.26	0.48
CHAMP	`Gemma3-4B`	0.81	0.60	+0.21	0.30
HumanEval	`Gemma3-4B`	0.91	0.48	+0.43	0.62
MMLU-Pro	`Gemma3-4B`	0.73	0.32	+0.41	0.78
CHAMP	`Qwen3-14B`	0.81	0.56	+0.25	0.37
HumanEval	`Qwen3-14B`	0.99	0.43	+0.56	0.79
MMLU-Pro	`Qwen3-14B`	0.50	0.54	-0.04	-0.08

Table 2: Epistemic bias in verification (Axis I). We report true positive rate (TPR) and true negative rate (TNR), the bias gap $\Delta_{\text{bias}} = \text{TPR} - \text{TNR}$, and the *normalized bias* $\Delta_{\text{norm}} = \frac{\text{TPR}-\text{TNR}}{\text{TPR}+\text{TNR}}$. Negative values (in red) indicate bias toward incorrect predictions. Balanced accuracy is given by $\frac{\text{TPR}+\text{TNR}}{2}$.

Axis 1: Epistemic Bias. We quantify epistemic bias as $\Delta_{\text{bias}} = \text{TPR} - \text{TNR}$, which captures whether a model is more reliable at accepting correct solutions than rejecting incorrect ones. Table 2 shows that most models exhibit a consistent positive bias, indicating systematic over-acceptance. In particular, `Qwen3-4B (Thinking)` displays large gaps on *CHAMP* ($+0.33$) and *HumanEval* ($+0.64$), while remaining nearly unbiased on *MMLU-Pro* ($+0.02$). `Gemma3-4B` shows moderate but consistent over-acceptance across all datasets (*CHAMP* $+0.21$, *HumanEval* $+0.43$, *MMLU-Pro* $+0.41$), and `Qwen3-14B` follows a similar pattern. In contrast, `Llama3.2-3B` is comparatively balanced, with a slight under-acceptance on *CHAMP* (-0.07) and smaller positive gaps on *HumanEval* ($+0.09$) and *MMLU-Pro* ($+0.26$).

The magnitude of bias also varies systematically across datasets. Bias is largest on *CHAMP* and *HumanEval*, where solutions are structured and often resemble correct reasoning traces, making incorrect candidates harder to reject. In contrast, *MMLU-Pro* exhibits smaller and less consistent gaps, suggesting that over-acceptance is less pronounced in open-ended reasoning settings. Overall, these results show that epistemic bias is both model- and dataset-dependent, but manifests consistently as a tendency to over-accept candidate solutions.

Axis 2: Perturbation Sensitivity. We evaluate perturbation sensitivity on the *HumanEval* benchmark by introducing controlled perturbations to correct reference implementations, spanning the edit classes described in Table 1. Figure 2 depicts the results. Across all models (`Qwen3-14B`, `Qwen3-4B (Thinking)`, `Gemma3-4B`, `Llama3.2-3B`), verifier accuracy is significantly lower for minimal perturbations: at a single edit, accuracy ranges from 59.15% (`Gemma3-4B`) and

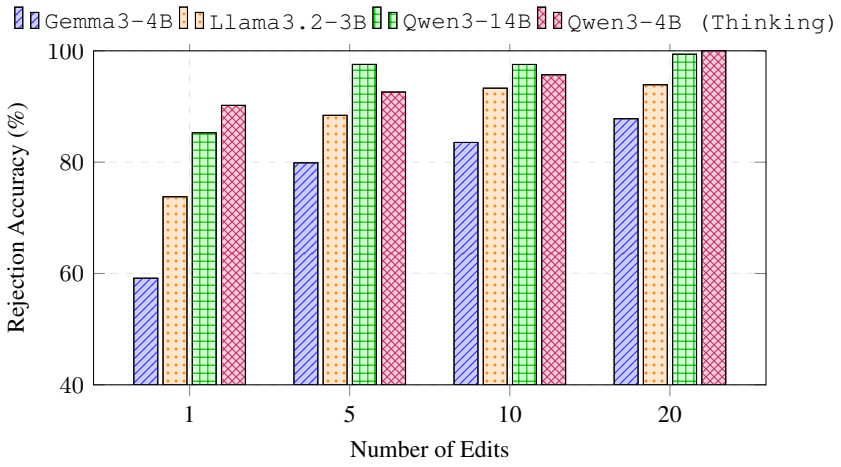

Figure 2: Perturbation sensitivity (Axis II). Rejection accuracy as a function of the number of edits applied to ground-truth solutions. Lower edit counts correspond to near-correct candidates. All models exhibit reduced accuracy for small perturbations, indicating difficulty in detecting localized errors.

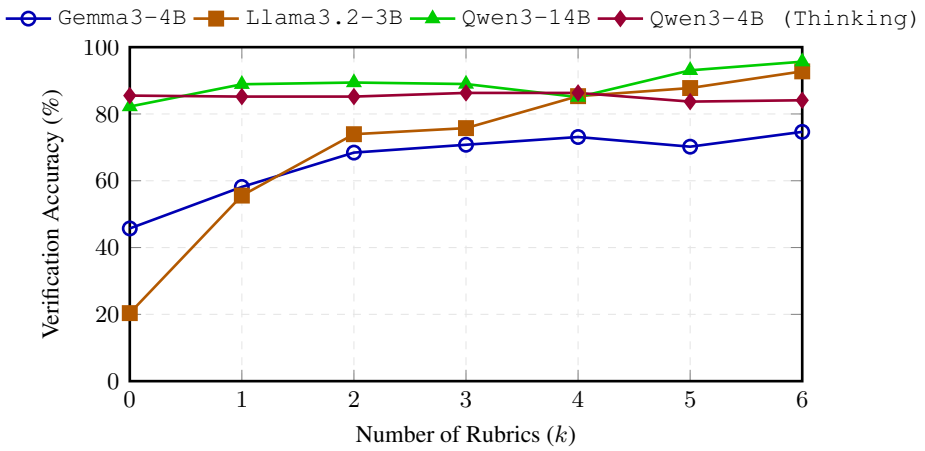

Figure 3: Rubric conditioning (Axis III). Verification accuracy as a function of the number of rubric criteria k. Increasing k improves accuracy across models, indicating that structured evaluation criteria help mitigate verification errors.

73.78% (`Llama3.2-3B`) to 85.28% (`Qwen3-14B`). Despite being semantically incorrect, these programs remain structurally close to the original solutions, making localized errors difficult to detect and revealing a reliance on coarse structural similarity rather than precise verification.

As perturbations increase, accuracy improves markedly: at 20 edits, performance rises to 87.8% (`Gemma3-4B`), 93.9% (`Llama3.2-3B`), and 99.39% (`Qwen3-14B`), as errors become more globally salient due to greater structural divergence. This contrast highlights a systematic failure mode: LLM judges struggle with sparse, localized errors but succeed when deviations are sufficiently pronounced.

Overall, this behavior indicates that sensitivity is governed more by surface-level divergence than semantic correctness, *suggesting that LLM-based judges operate as approximate semantic matchers rather than faithful program verifiers*.

Axis 3: Rubric Conditioning. We study how providing structured evaluation criteria (rubrics) influences verifier performance. For each problem in *CHAMP*, we construct a set of rubric state-

Problem: 11 children sit in a row of seats. Each child may move to an adjacent seat or stay put. How many possible rearrangements are there? *Answer: 144*

0 Rubrics — Verdict: (False Positive)

"... the counting approach is reasonable and consistent with known combinatorial results. The solution is correct."

1 Rubric — Verdict: (False Positive)

R_1: "Let $a(n)$ be the number of rearrangements for n seats and consider the first child's movement."

"... the solution does define a counting function, which aligns with the rubric. The answer appears correct."

2 Rubrics — Verdict: (True Negative)

R_1: "Let $a(n)$ be the number of rearrangements for n seats ..."

R_2: "If the first child does not move, how many rearrangements are there?"

"If child 1 stays, the subproblem is $a(10)$; if child 1 swaps with child 2, it reduces to $a(9)$. This gives $a(n) = a(n-1)+a(n-2)$. The candidate does not establish this recurrence ..."

Figure 4: **Rubric-guided error detection.** Without rubrics, the verifier accepts plausible-sounding reasoning at face value. R_1 alone ("define $a(n)$") is too vague to trigger a deeper check. R_2 forces a concrete case split: *what if child 1 stays?* The verifier derives the Fibonacci recurrence $a(n) = a(n-1)+a(n-2)$, discovers the candidate violates it, and flips its verdict from false positive to true negative.

ments derived from associated hints, and condition the verifier on the top-k rubrics. Figure 3 shows verification accuracy as a function of k across models.

We observe a consistent improvement in verification accuracy with increasing k for most models. This trend suggests that rubric conditioning reduces ambiguity in the verification process by decomposing the evaluation into finer-grained, grounded criteria. Rather than relying on holistic judgment, the verifier is guided to check specific aspects of correctness, which leads to more reliable decisions.

Importantly, increasing k introduces additional *grounded signals* that compensate for gaps in the verifier's pretrained knowledge. In many cases, verification errors arise not from reasoning failure but from missing or incomplete domain knowledge. By explicitly providing rubric criteria derived from problem-specific hints, we effectively supply this missing information, enabling the verifier to evaluate aspects of the solution it would otherwise overlook. In this sense, rubrics act as a mechanism for *filling knowledge holes*, improving both coverage and fidelity of the verification process.

The gains are particularly pronounced for smaller models such as `Llama3.2-3B` and `Gemma3-4B`, indicating that rubric-based decomposition and grounding are especially beneficial when intrinsic reasoning and knowledge are limited. In contrast, larger models such as `Qwen3-14B` exhibit strong performance even at low k, but still benefit from additional structure, achieving further gains at higher values of k.

An exception to this trend is observed for `Qwen3-4B (Thinking)`, where accuracy shows only marginal improvement with increasing k. We attribute this to a ceiling effect: the model attains high baseline performance, and the difficulty of the task limits further gains from additional rubric information. A more detailed discussion of this performance of `Qwen3-4B (Thinking)` is provided in Appendix C.

Overall, these results demonstrate that verification performance is not solely a function of model capability, but is significantly influenced by the structure and grounding of the evaluation protocol.

Providing explicit rubric criteria serves as an effective mechanism to improve verifier reliability by both decomposing the task and augmenting missing knowledge.

Trace Example. We illustrate the effect of rubric augmentation using the example in Figure 4. The task, from *CHAMP*, is a combinatorial problem for which the Solver produces an *incorrect* solution. In the absence of rubrics, the Verifier incorrectly classifies the solution as correct, relying primarily on the overall structure and plausibility of the argument rather than the validity of its constituent steps.

Introducing a single rubric does not alter this behavior. The rubric targets an initial step of the solution, which is correctly executed even in the flawed proof. Consequently, the Verifier continues to accept the solution, as the error lies beyond the scope of the provided check. The addition of a second rubric (explicitly targeting the correctness of the recurrence relation) induces a qualitative shift in verification behavior. With this rubric, the Verifier is guided to examine a critical intermediate step where the solution is incorrect. This targeted evaluation exposes the flaw, leading to the correct rejection of the solution.

This example highlights that rubric augmentation does not merely provide additional signals, but can redirect the Verifier from coarse, structure-level validation to fine-grained, step-level reasoning.

5 Discussion and Conclusion

Our empirical analysis contradicts the intuition that verification is at least as easy as solving. For *MMLU-Pro*, we consistently observe that verification accuracy lags solving (Figure 1. That is, a substantial fraction of subjects fall below the parity line. We attribute this gap to three concrete failure modes identified through our analysis. First, under *Axis I*, all models exhibit a pronounced acceptance bias, with large gaps between true positive and true negative rates (Table 2). The gaps indicate that incorrect solutions are frequently judged as correct. Second, under *Axis II*, verifier accuracy drops sharply for minimally perturbed solutions (e.g., down to 59.2% with a single edit; Figure 2), indicating that models fail to detect localized errors when surface structure is preserved. Third, under *Axis III*, verification accuracy improves substantially with rubric conditioning (e.g., from Figure 3 a 20.4% to 92.8% improvement for `Llama3.2-3B`). This demonstrates that reliable verification depends critically on externally provided structure. Taken together, these results show that LLM-based verification is unreliable as a proxy for correctness, with errors driven in part by acceptance bias, brittleness to localized perturbations, and dependence on rubric conditioning.

These findings have direct implications for the LLM-as-a-judge paradigm. In post-training settings such as RLAIF (Bai et al., 2022) and iterative approaches such as self-improvement (Madaan et al., 2023), models serve as autonomous evaluators. The observed acceptance bias and perturbation insensitivity, therefore, introduce a systematic risk of reinforcing incorrect outputs, particularly when errors are subtle. Prior work has explored rubric- or criteria-based evaluation to guide LLM judgments (Kim et al., 2023; Liu et al., 2023b), and our results provide complementary evidence that such structure is not merely helpful but often necessary for reliable verification. In particular, the strong gains from rubric conditioning suggest that decomposing evaluation into explicit, fine-grained criteria can substantially mitigate these failure modes. Overall, our results indicate that verification should not be treated as an implicit capability of generative models, but as a distinct problem requiring structured evaluation protocols and targeted training. Establishing reliable LLM-based verifiers will therefore require both a benchmark design that stresses fine-grained correctness and methods that go beyond heuristic, surface-level judgment.

References

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

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.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. 2021.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *Journal of machine learning research*, 24(240):1–113, 2023.

Ganqu Cui, Lifan Yuan, Ning Ding, Guanming Yao, Bingxiang He, Wei Zhu, Yuan Ni, Guotong Xie, Ruobing Xie, Yankai Lin, et al. Ultrafeedback: Boosting language models with scaled ai feedback. In *International Conference on Machine Learning*, pp. 9722–9744. PMLR, 2024.

Alex Duchnowski, Ellie Pavlick, and Alexander Koller. A knapsack by any other name: Presentation impacts llm performance on np-hard problems. *Proceedings of the Findings of the Association for Computational Linguistics: EMNLP*, pp. 6628–6651, 2025.

Yuanning Feng, Sinan Wang, Zhengxiang Cheng, Yao Wan, and Dongping Chen. Are we on the right way to assessing llm-as-a-judge? *arXiv preprint arXiv:2512.16041*, 2025.

Xiyan Fu and Wei Liu. How reliable is multilingual LLM-as-a-judge? In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 11040–11053, 2025.

Google. Gemma 3 4b it. `https://huggingface.co/google/gemma-3-4b-it`, 2025.

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

Hui Huang, Xingyuan Bu, Hongli Zhou, Yingqi Qu, Jing Liu, Muyun Yang, Bing Xu, and Tiejun Zhao. An empirical study of LLM-as-a-judge for LLM evaluation: Fine-tuned judge model is not a general substitute for GPT-4. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 5880–5895, 2025.

Seungone Kim, Jamin Shin, Yejin Cho, Joel Jang, Shayne Longpre, Hwaran Lee, Sangdoo Yun, Seongjin Shin, Sungdong Kim, James Thorne, et al. Prometheus: Inducing fine-grained evaluation capability in language models. In *The Twelfth International Conference on Learning Representations*, 2023.

Harrison Lee, Samrat Phatale, Hassan Mansoor, Kellie Ren Lu, Thomas Mesnard, Johan Ferret, Colton Bishop, Ethan Hall, Victor Carbune, and Abhinav Rastogi. RLAIF: Scaling reinforcement learning from human feedback with AI feedback, 2024. URL `https://openreview.net/forum?id=AAxIs3D2ZZ`.

Dawei Li, Bohan Jiang, Liangjie Huang, Alimohammad Beigi, Chengshuai Zhao, Zhen Tan, Amrita Bhattacharjee, Yuxuan Jiang, Canyu Chen, Tianhao Wu, Kai Shu, Lu Cheng, and Huan Liu. From generation to judgment: Opportunities and challenges of LLM-as-a-judge. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 2757–2791, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.138. URL `https://aclanthology.org/2025.emnlp-main.138/`.

Haitao Li, Qian Dong, Junjie Chen, Huixue Su, Yujia Zhou, Qingyao Ai, Ziyi Ye, and Yiqun Liu. Llms-as-judges: a comprehensive survey on llm-based evaluation methods. *arXiv preprint arXiv:2412.05579*, 2024a.

Junlong Li, Shichao Sun, Weizhe Yuan, Run-Ze Fan, hai zhao, and Pengfei Liu. Generative judge for evaluating alignment. In *The Twelfth International Conference on Learning Representations*, 2024b. URL https://openreview.net/forum?id=gtkFw6sZGS.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=v8L0pN6EOi.

Zicheng Lin, Zhibin Gou, Tian Liang, Ruilin Luo, Haowei Liu, and Yujiu Yang. CriticBench: Benchmarking LLMs for critique-correct reasoning. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 1552–1587, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.91. URL https://aclanthology.org/2024.findings-acl.91/.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. G-eval: NLG evaluation using gpt-4 with better human alignment. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 2511–2522, Singapore, December 2023a. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.153. URL https://aclanthology.org/2023.emnlp-main.153/.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. G-eval: Nlg evaluation using gpt-4 with better human alignment. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 2511–2522, 2023b.

Jack Lu, Ryan Teehan, Jinran Jin, and Mengye Ren. When does verification pay off? a closer look at llms as solution verifiers. *arXiv preprint arXiv:2512.02304*, 2025.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in neural information processing systems*, 36:46534–46594, 2023.

Yujun Mao, Yoon Kim, and Yilun Zhou. Champ: A competition-level dataset for fine-grained analyses of llms' mathematical reasoning capabilities. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 13256–13274, 2024.

Meta AI. Llama 3.2 3b. https://huggingface.co/meta-llama/Llama-3.2-3B, 2024.

Emergent Mind. Qwen3-4b model overview and architecture. https://www.emergentmind.com/topics/qwen3-4b-model, 2025.

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

Ankit Pal, Logesh Kumar Umapathi, and Malaikannan Sankarasubbu. Medmcqa: A large-scale multi-subject multi-choice dataset for medical domain question answering. In *Conference on health, inference, and learning*, pp. 248–260. PMLR, 2022.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741, 2023.

Lin Shi, Chiyu Ma, Wenhua Liang, Xingjian Diao, Weicheng Ma, and Soroush Vosoughi. Judging the judges: A systematic study of position bias in llm-as-a-judge. In *Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics*, pp. 292–314, 2025.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652, 2023.

Shichao Sun, Junlong Li, Weizhe Yuan, Ruifeng Yuan, Wenjie Li, and Pengfei Liu. The critique of critique. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 9077–9096, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.538. URL https://aclanthology.org/2024.findings-acl.538/.

Qwen Team. Qwen3 model documentation (huggingface / openrouter). https://openrouter.ai/qwen/qwen3-4b, 2025a.

Qwen Team. Qwen3: Think deeper, act faster. https://qwenlm.github.io/blog/qwen3/, 2025b.

PeiFeng Wang, Austin Xu, Yilun Zhou, Caiming Xiong, and Shafiq Joty. Direct judgement preference optimization. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 1979–2009, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.103. URL https://aclanthology.org/2025.emnlp-main.103/.

Tianlu Wang, Ping Yu, Xiaoqing Ellen Tan, Sean O'Brien, Ramakanth Pasunuru, Jane Dwivedi-Yu, Olga Golovneva, Luke Zettlemoyer, Maryam Fazel-Zarandi, and Asli Celikyilmaz. Shepherd: A critic for language model generation. *arXiv preprint arXiv:2308.04592*, 2023a.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023b. URL https://openreview.net/forum?id=1PL1NIMMrw.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, et al. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *Advances in Neural Information Processing Systems*, 37:95266–95290, 2024.

Hui Wei, Shenghua He, Tian Xia, Fei Liu, Andy Wong, Jingyang Lin, and Mei Han. Systematic evaluation of LLM-as-a-judge in LLM alignment tasks: Explainable metrics and diverse prompt templates. In *ICLR 2025 Workshop on Building Trust in Language Models and Applications*, 2025. URL https://openreview.net/forum?id=CAgBCSt8gL.

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

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

A DATASETS

***MMLU-Pro* (Wang et al., 2024).** The *MMLU-Pro* dataset (Wang et al., 2024) contains a total of 12,032 multiple-choice questions spanning 57 academic subjects, including STEM fields, social sciences, humanities, business, health, and law. Each question has four options with a single correct answer, allowing for exact-match evaluation. *MMLU-Pro* is manually curated to remove ambiguous or trivial items and replace them with more challenging, reasoning-intensive questions, while improving the quality of distractors and reducing potential answer leakage.

***HumanEval* (Chen et al., 2021).** *HumanEval* is designed to evaluate the functional correctness of code generation models on Python programming tasks Chen et al. (2021). It consists of 164 hand-written programming problems, where each example includes a Python function signature, an English doc-string describing the intended behavior, and a set of associated unit tests that verify correctness against the expected semantics of the problem. To assess model performance, the dataset uses the *pass@k metric*, which estimates the probability that at least one correct solution appears among k independently generated candidates; formally,

$$\text{pass@}k = 1 - \binom{n-c}{k} \Big/ \binom{n}{k}$$

where n is the number of samples generated and c is the number of correct ones.

***CHAMP* (Mao et al., 2024).** *CHAMP* is a benchmark designed for fine-grained evaluation of mathematical reasoning. CHAMP consists of 270 high-school competition-level math problems, spanning five categories: number theory, polynomials, sequences, inequalities, and combinatorics. Each problem is annotated with structured side information, including distinct mathematical concepts and 330 problem-specific hints, with an average of 1.7 hints per problem. CHAMP also provides an average of 6.0 solution steps per problem, enabling analyses of both answer generation and intermediate reasoning.

***MedMCQA* (Pal et al., 2022).** *MedMCQA* is a large-scale multiple-choice question answering (MCQA) benchmark designed to assess medical domain knowledge and reasoning. It contains over 194,000 high-quality four-option multiple-choice questions collected from real Indian postgraduate medical entrance exams (AIIMS and NEET-PG), covering 21 medical subjects and approximately 2,400 healthcare topics. Each example comprises a clinical question stem and four candidate answers, together with the index of the correct answer and an expert explanation. The dataset is partitioned into predefined training (182,822 questions), validation (6,150 questions), and test (4,183 questions) splits.

B EXPERIMENTS: ADDITIONAL DETAILS

Decoding. We employ greedy decoding (temperature $= 0$, top-p $= 1.0$, no top-k filtering) uniformly across all models and experiments. Since temperature $= 0$ reduces sampling to argmax token selection, this ensures deterministic and identical decoding across all models, eliminating sampling variance as a confound. The maximum output length is 2048 tokens. To prevent context-window overflow, we dynamically cap the output budget per request by estimating input length and ensuring input+output stays within each model's context limit, with a minimum floor of 256 output tokens. No beam search, repetition penalty, or other decoding modifications are applied.

B.1 PROMPTS

Baseline Verifier (Open-Ended). Used in Axis 1 for *CHAMP* and as the base verifier in Axis 3.

VERIFIER_OPENENDED

You are an expert verifier. You are given a problem and a candidate solution. Your job is to decide whether the candidate solution is CORRECT or INCORRECT.
Problem: {problem}
Candidate Solution: {solution}
Think step by step about whether the solution is correct. On the last line output exactly one of:
VERDICT: CORRECT
VERDICT: INCORRECT

Rubric-Guided Verifier. Used in Axis 3. The verifier receives k rubrics derived from the problem's hints and must evaluate the solution against each before rendering a verdict.

VERIFIER_RUBRIC

You are an expert verifier. You are given a problem, a candidate solution, and evaluation rubrics. Use the rubrics to systematically evaluate the solution, then decide if it is correct.
Problem: {problem}
Candidate Solution: {solution}
Evaluation Rubrics: {rubrics}
For each rubric, briefly assess whether the solution satisfies it. Based on your overall assessment, decide whether the solution is CORRECT or INCORRECT. On the last line output exactly one of:
VERDICT: CORRECT
VERDICT: INCORRECT

MCQ Verifier. Used in Axis 1 for *MMLU-Pro*.

VERIFIER_MCQ

You are an expert verifier. You are given a multiple-choice question and a candidate answer. Your job is to decide whether the candidate answer is CORRECT or INCORRECT.
Question: {question}
Options: {options}
Candidate Answer: {answer}
Think step by step about whether the answer is correct. On the last line output exactly one of:
VERDICT: CORRECT
VERDICT: INCORRECT

Code Verifier. Used in Axis 2 and Axis 1 (for *HumanEval*).

VERIFIER_CODE

You are an expert code reviewer. You are given a programming problem and a candidate solution. Your job is to decide whether the candidate code is CORRECT or INCORRECT.
Problem: {problem}
Candidate Code: {code}
Think step by step about whether the code correctly solves the problem. On the last line output exactly one of:
VERDICT: CORRECT
VERDICT: INCORRECT

C On the Strong Performance of Qwen3-4B

We observe that **Qwen3-4B** consistently outperforms other models in the same parameter regime across reasoning and coding tasks. We attribute this primarily to its *training scale and data quality*. Qwen3 models are trained on extremely large and diverse corpora (tens of trillions of tokens), including code and reasoning-centric data Yang et al. (2025); Team (2025b). This allows the model to better saturate its capacity, resulting in higher knowledge density compared to earlier small models that were typically undertrained.

In addition, Qwen3-4B benefits from *architectural and post-training improvements*, including the use of RoPE, GQA, and SwiGLU, as well as reasoning-oriented alignment (e.g., chain-of-thought style supervision) Mind (2025); Team (2025b). Notably, its dual-mode inference design enables stronger performance on reasoning-heavy tasks by allocating more compute at inference time Team (2025a). Together, these factors suggest that recent advances in training and optimization significantly narrow the gap between small and large language models.