JUDGE NOISE, NOT DIFFICULTY: DIAGNOSING WEAK OPEN-SOURCE VERIFIERS IN RLVR

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning from verifiable rewards (RLVR) increasingly replaces the gold verifier with a learned or zero-shot LLM *judge*, especially in verify-hard domains such as multi-hop question answering where exact-match signals are brittle. A common implicit assumption is that such judges are *reliable enough*: that their per-example verdicts reflect true answer correctness, so that the group statistics driving RLVR (e.g., group-relative advantage in GRPO) are estimated on signal rather than noise. We show this assumption fails badly for the weak open-source judges that are actually used in practice. On a controlled multi-hop QA verification suite, three weak base judges (Qwen3-8B, Qwen2.5-7B, Qwen3-4B) exhibit per-prompt verdict instability of 39–66%: given the same (question, candidate) pair, the same judge flips between CORRECT and WRONG across repeated samples. A strong commercial judge is, by contrast, essentially stable (0% flips) on the same prompts, confirming that the instability is a property of *weak* verifiers and not of the task. Crucially, this instability is not a parsing artifact: one judge parses a clean verdict 99% of the time yet still flips on 66% of prompts. Because variance-driven rollout allocation cannot distinguish judge noise from true difficulty, it mis-allocates sampling budget toward judge-unstable prompts. We further report a negative control: an RLVR policy trained against a same-source judge yields no robust gain even under that same judge (mean $+2.6$ points, standard deviation 2.0 points across repeated scoring, spanning zero), because single-run judge scores are themselves too noisy to trust. Together these results argue that judge-noise diagnostics should be a standard prerequisite before adopting an open-source verifier as an RLVR reward.

1 INTRODUCTION

Reinforcement learning from verifiable rewards (RLVR) has become a dominant recipe for improving the reasoning ability of large language models (Yang et al., 2024; DeepSeek-AI et al., 2025). In its cleanest form, the reward is a deterministic checker: a math answer is compared against a known solution, or a program is run against unit tests. In these domains the reward is *verifiable* by construction, and the group statistics that RLVR algorithms such as GRPO (Shao et al., 2024) rely on—per-prompt means and variances over sampled rollouts—are estimated on a clean signal.

Many important domains, however, are *verify-hard*: there is no cheap deterministic checker. Multi-hop question answering is a canonical example, where a semantically correct answer can differ from the reference string, so exact match is brittle (Trivedi et al., 2022; Yang et al., 2018). The now-common workaround is to replace the gold checker with an LLM *judge*: either a strong commercial model prompted zero-shot, or a smaller open-source model that can be served cheaply alongside training (Zheng et al., 2023; Li et al., 2025). An implicit assumption underlies this move: that the judge is *reliable enough*—that its per-example verdicts track true answer correctness closely enough that the RLVR group statistics remain meaningful.

We argue this assumption is unsafe precisely for the judges practitioners can afford to run at training scale. We study weak open-source base judges on a controlled multi-hop QA verification task and

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

ask a simple question: given the *same* (question, candidate) pair, how often does the *same* judge change its verdict across independent samples? Our findings are:

- **Weak judges are highly self-inconsistent.** Three weak base judges (Qwen3-8B, Qwen2.5-7B, Qwen3-4B) flip their CORRECT/WRONG verdict on 39–66% of prompts under repeated sampling, with per-prompt Bernoulli variance up to 0.17 (Section 4). A strong commercial judge flips on 0% of the same prompts, showing the instability is a property of weak verifiers, not of the task.
- **The noise is not a parsing artifact.** One judge emits a well-formed verdict on 99% of samples yet still flips on 66% of prompts: it reliably says something, but what it says swings between correct and wrong.
- **Judge noise masquerades as difficulty.** Because variance-driven rollout allocation treats high per-prompt verdict variance as high difficulty, it mis-allocates sampling budget toward prompts that are merely judge-unstable, not genuinely hard.
- **A negative control on training.** An RLVR policy trained against a same-source judge shows no robust improvement even under that same judge (mean $+2.6$ points, s.d. 2.0 points over repeated scoring, spanning zero; McNemar $p{=}0.77$), because single-run judge scores are themselves too noisy to certify a gain (Section 4).

Our contribution is a diagnostic, not a new training algorithm. We give a cheap, controlled protocol to quantify judge self-inconsistency, show it is severe for the open-source judges used in practice, and argue that such a diagnostic should be run *before* an open-source verifier is adopted as an RLVR reward. We are explicit about scope: we measure the judge-noise *prerequisite* for reward misallocation and report a training-side negative control, but we do not claim an end-to-end causal chain from judge noise to final task accuracy (Section 5).

2 Related Work

RL alignment and verifiable rewards. Reinforcement learning has become central to aligning and improving LLMs, from RLHF on human preferences (Ouyang et al., 2022; Stiennon et al., 2022) and its offline counterpart DPO (Rafailov et al., 2024) to reinforcement learning from verifiable rewards (RLVR), which trains policies against automatically checkable rewards and underlies recent reasoning-focused systems (Yang et al., 2024; DeepSeek-AI et al., 2025; Lambert et al., 2025). GRPO and related group-relative methods estimate a baseline from multiple sampled rollouts per prompt, so the per-prompt reward *distribution*—not just its mean—directly shapes the advantage and hence the gradient (Shao et al., 2024; Schulman et al., 2017). Verifiable signals range from outcome checkers to process-level supervision (Uesato et al., 2022; Lightman et al., 2023) and increasingly to model-based or generative verifiers (Yuan et al., 2025; Jiang et al., 2025; Feng et al., 2025; Aksitov et al., 2023). This makes RLVR sensitive to any structure in the reward's variance, which is exactly what a noisy judge injects.

LLMs as judges. Using an LLM to score another model's output is now standard for open-ended and verify-hard tasks (Zheng et al., 2023; Li et al., 2025), spanning reference-based scorers such as G-Eval, PandaLM, and Prometheus (Liu et al., 2023; Wang et al., 2024; Kim et al., 2024). Known failure modes include position and verbosity biases and self-preference (Wang et al., 2023a; Saito et al., 2023; Panickssery et al., 2024), and even strong judges show only limited rating consistency (Hackl et al., 2023). Most of this literature studies *systematic* bias in a single pass; we instead quantify *stochastic* self-inconsistency—repeated-sampling verdict variance on a fixed input—which is the property that corrupts RLVR group statistics.

Reward noise, hacking, and robustness. Misspecified or exploitable rewards are a long-standing concern, from concrete AI-safety worries (Amodei et al., 2016) to formal characterizations of reward hacking (Skalse et al., 2025), reward-model overoptimization (Gao et al., 2022), and ensemble mitigations that reduce but do not eliminate it (Eisenstein et al., 2024). Within RLVR specifically, recent work shows that even spurious or partially incorrect reward signals can still drive gains (Shao et al., 2026), and a related line argues that LLMs are surprisingly robust to reward noise (Lv et al., 2025). We test this robustness claim in a verify-hard setting and find that the relevant judges are

noisy enough, and in a prompt-correlated way, that variance-driven allocation is misled; robustness to i.i.d. label flips does not imply robustness to structured judge noise.

Multi-hop QA verification. Answer-matching evaluation for QA is well known to be brittle: exact match against a single reference under-counts semantically correct answers, a problem already visible on span-based and open-domain QA (Rajpurkar et al., 2016; Alberti et al., 2019) and compounded by generation hallucination (Ji et al., 2024). Multi-hop QA benchmarks such as MuSiQue, HotpotQA, and 2WikiMultihopQA (Trivedi et al., 2022; Yang et al., 2018; Ho et al., 2020) are especially verify-hard because correct answers vary in surface form, motivating LLM judges and making them a natural testbed for judge-noise diagnostics.

3 METHOD: A JUDGE-NOISE DIAGNOSTIC

3.1 SETUP

We work in a verify-hard domain: multi-hop question answering on MuSiQue (Trivedi et al., 2022), where each example provides a question, a candidate answer, and a gold reference (with aliases). A *judge* J is an LLM prompted to decide whether the candidate answers the question; it emits free-form reasoning followed by a final line VERDICT: CORRECT or VERDICT: WRONG, which we parse into a binary label $v \in \{\text{CORRECT}, \text{WRONG}\}$ (or *unparsed*).

3.2 JUDGE SELF-INCONSISTENCY

The quantity of interest is how much a judge's verdict on a *fixed* input is driven by its own sampling noise rather than by true correctness. For each prompt i (a fixed (question, candidate) pair) we sample the judge k times at temperature τ and collect parsed verdicts. Let $\hat{p}_i$ be the fraction of samples labelled CORRECT. We report three statistics:

- **Per-prompt Bernoulli variance** $\hat{p}_i(1-\hat{p}_i)$, averaged over prompts. This is exactly the per-prompt reward variance that a variance-driven allocator would read as "difficulty."
- **Has-flip rate**: the fraction of prompts with $0 < \hat{p}_i < 1$, i.e. at least one disagreement among the k samples.
- **Unstable-flip rate**: the fraction of prompts whose verdict distribution is close to a coin flip ($\hat{p}_i(1-\hat{p}_i) \geq 0.1875$, i.e. a minority verdict on at least two of k=8 samples), our threshold for "the verdict is dominated by noise."

A judge whose verdicts reflect true correctness should have near-zero variance on most prompts; large average variance means the reward signal fed to RLVR is, on many prompts, noise. This is the flip side of test-time aggregation: whereas self-consistency and best-of-n style methods deliberately sample many times to *reduce* answer variance (Wang et al., 2023b; Snell et al., 2024), an RLVR pipeline typically queries the judge *once* per rollout, so any such verdict variance is passed through undiluted as reward noise.

3.3 WHY THIS CORRUPTS VARIANCE-DRIVEN ALLOCATION

GRPO-style RLVR samples multiple rollouts per prompt and forms advantages from the per-prompt reward distribution. Allocation heuristics that spend more budget on high-variance prompts implicitly assume variance tracks difficulty. If instead a large share of per-prompt variance comes from judge self-inconsistency, the allocator preferentially samples *judge-unstable* prompts, spending budget where the reward is least informative. Section 4 measures the size of this judge-noise component directly.

3.4 TRAINING-SIDE NEGATIVE CONTROL

To probe whether the judge noise couples to actual training, we additionally train a policy with RLVR using a fixed same-source judge as the reward and score its answers under that same judge,

a strong exact/alias gold matcher, and (as a gate) a cross-source judge. Because we find the same-source gain is within the judge's own scoring noise, we report this as a *negative control* rather than a positive training result; the exact protocol and numbers are in Section 4.

3.5 JUDGE PROMPT AND PARSING PROTOCOL

To make the diagnostic reproducible and to isolate verdict noise from prompt engineering, all judges receive an identical zero-shot template. The template presents the question, the candidate answer, and the gold reference with its known aliases, and instructs the model to reason briefly and then emit a final line of the exact form VERDICT: CORRECT or VERDICT: WRONG. We deliberately expose the gold reference to the judge: this is the most favourable setting for the judge (it is graded against the answer key rather than asked to verify open-endedly), so any instability we observe is a *lower bound* on the noise a judge would exhibit in a harder, reference-free reward setting.

Parsing is intentionally lenient. We scan the completion for the last occurrence of a verdict token, accepting common surface variants (case, surrounding punctuation, and the synonyms "yes"/"no", "true"/"false") before falling back to *unparsed*. A sample is counted as *unparsed* only when no verdict token can be recovered at all. Crucially, the parsed-fraction column in Table 1 therefore measures whether the judge *committed* to a verdict, not whether we could decode a sloppily formatted one; a high parsed fraction with a high flip rate (as for Qwen2.5-7B) means the judge confidently emits contradictory verdicts across samples, which is the failure mode most damaging to an RLVR reward.

3.6 EXPERIMENTAL SETUP

Judges. We compare four judges: a strong commercial API judge (doubao-seed-2-0-pro), retained as an adverse control because a stable strong judge is the hardest case for our hypothesis, and three weak open-source base judges spanning two model families and three sizes (Qwen3-8B, Qwen2.5-7B, Qwen3-4B). All open-source judges are *base* checkpoints served locally, matching the cost regime in which practitioners actually deploy a training-time verifier.

Sampling. Each judge is queried $k{=}8$ times per prompt at temperature $\tau{=}0.7$, the same sampling regime an RLVR pipeline would use when the judge is invoked once per rollout. We hold the (question, candidate) pair fixed across the k samples so that any variation in the verdict is attributable to the judge's own stochasticity rather than to a changing input.

Data. Prompts are drawn from the MuSiQue multi-hop QA verification suite (Trivedi et al., 2022); candidates include both gold-correct and gold-incorrect answers so that the stratified analysis of Section 4.2 is well defined on both classes. We report the number of scored prompts per judge in Table 1; differences in prompt counts across judges reflect only how many prompts yielded at least two parsed samples (the minimum needed to define a flip).

Training (negative control). For the training-side control we run GRPO on a Qwen3-8B policy for MuSiQue answering, using a fixed same-source judge as the sole reward, and evaluate the base and trained policies on a held-out set of 500 items under (i) the same-source judge and (ii) a gold exact/alias matcher. All same-source evaluations are repeated three times to expose the judge's own scoring noise; hyperparameters and the exact evaluation protocol are listed in Appendix B.

4 EXPERIMENTS

We first quantify judge self-inconsistency across four verifiers (Section 4.1), then dissect where the noise comes from and how many samples are needed to suppress it (Sections 4.2 and 4.3), and finally report a training-side negative control (Section 4.4).

Protocol summary. Unless noted otherwise, every judge is queried $k{=}8$ times per prompt at temperature $\tau{=}0.7$ on the MuSiQue verification suite, with the (question, candidate) pair held fixed across samples so that all verdict variation is attributable to the judge's own stochasticity (Section 3.6). We report a prompt as *scored* for a judge once it yields at least two parsed verdicts, the minimum needed to define a flip. Because judge decoding is stochastic even at low temperature under continuous-batched serving, every aggregate we report is computed over the full sample set

Table 1: Judge self-inconsistency on MuSiQue verification (k=8, τ=0.7). "Mean var" is the average per-prompt Bernoulli variance; "unstable-flip" is the fraction of prompts dominated by noise ($\hat{p}(1-\hat{p}) \geq 0.1875$); "has-flip" is the fraction with any disagreement; "parsed" is the fraction of samples yielding a well-formed verdict. The strong commercial judge is stable; all three weak base judges are severely self-inconsistent, and parsing success does not explain it.

Judge	Prompts	Mean var	Unstable-flip	Has-flip	Parsed
doubao-seed-2-0-pro (strong API)	29	0.000	0.0%	0.0%	100%
Qwen3-8B base (RLVR base)	803	0.112	39.1%	57.4%	60.1%
Qwen2.5-7B base (cross-family)	944	0.167	**65.8%**	87.2%	98.8%
Qwen3-4B base (smaller)	815	0.120	44.3%	58.7%	57.2%

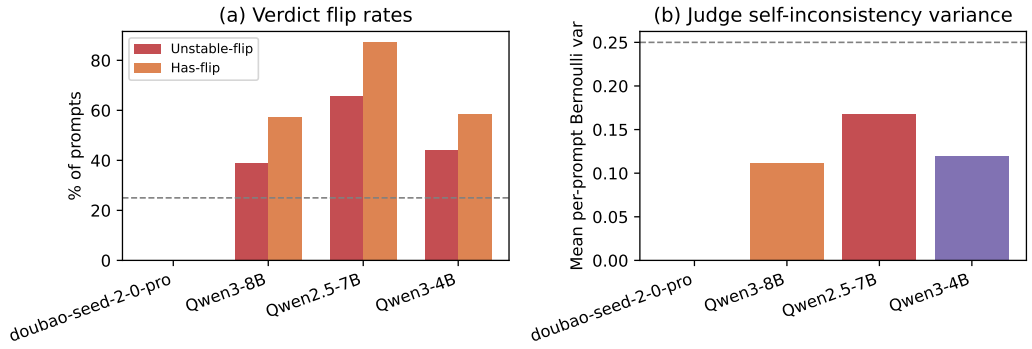

Figure 1: Verdict flip rates (a) and mean per-prompt Bernoulli variance (b) for the four judges. The dashed line marks a 25% noise-dominated threshold in (a) and the coin-flip maximum 0.25 in (b). Every weak open-source judge lies far above the noise threshold, while the strong commercial judge sits at zero.

rather than a single pass, and the training-side same-source evaluation of Section 4.4 is repeated three times and reported as mean±s.d. This protocol is deliberately cheap—a few thousand judge calls per verifier—so that it can be run as a pre-adoption check before an open-source judge is wired into an RLVR reward.

4.1 Judge self-inconsistency across verifiers

We evaluate four judges on the MuSiQue verification suite: one strong commercial API judge (doubao-seed-2-0-pro, held out as the most adverse control for our hypothesis) and three weak open-source base judges spanning two families and three sizes (Qwen3-8B, Qwen2.5-7B, Qwen3-4B). Each judge is sampled k=8 times per prompt at temperature τ=0.7; statistics are defined in Section 3. Table 1 and Figure 1 report the results.

Three observations follow. First, **judge strength decides the signal**: the strong commercial judge never flips on the sampled prompts, whereas every weak open-source judge flips on 39–66% of prompts (Figure 1a), all far above a 25% noise-dominated threshold. Real RLVR pipelines use exactly the weak, cheap-to-serve judges, so they inherit this instability. Second, **the noise is not a parsing artifact**: Qwen2.5-7B parses a clean verdict 98.8% of the time yet still flips on 65.8% of prompts—it reliably emits a verdict whose content swings between CORRECT and WRONG. Third, **the average per-prompt variance is large** (0.11–0.17, against a maximum of 0.25 for a pure coin flip; Figure 1b), so a variance-driven allocator reading this signal as difficulty would mis-allocate a large fraction of its budget to judge-unstable prompts. Figure 2 shows the full per-prompt variance distributions: the weak judges place substantial mass near the coin-flip maximum, whereas the strong judge concentrates at zero.

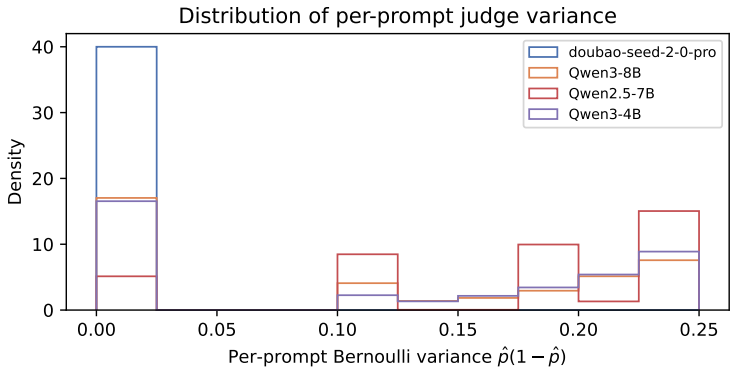

Figure 2: Distribution of per-prompt Bernoulli variance across judges. Weak base judges (orange/red/purple) place substantial mass near the coin-flip maximum 0.25; the strong API judge concentrates at zero (not visible because its mass is a spike at 0).

Table 2: Has-flip rate stratified by candidate ground truth. Weak judges flip on both correct and wrong candidates at similar rates, so the instability is generic verdict noise, not calibrated uncertainty on hard cases.

Judge	Correct-candidate flip	Wrong-candidate flip
Qwen3-8B base	55.6% (n=403)	59.2% (n=400)
Qwen2.5-7B base	87.9% (n=472)	86.4% (n=472)
Qwen3-4B base	55.9% (n=410)	61.5% (n=405)
doubao-seed-2-0-pro	0.0% (n=18)	0.0% (n=11)

4.2 Where does the noise come from?

A natural worry is that the instability is concentrated on genuinely ambiguous cases—e.g. only on wrong candidates that are "close" to correct. We stratify prompts by the ground-truth correctness of the candidate answer (*correct candidate* vs. *wrong candidate*) and recompute the has-flip rate; Table 2 and Figure 3 show the noise is present on *both* strata at similar magnitude for every weak judge. The verifier is not merely uncertain on hard negatives; it is unstable even when asked to confirm a truly correct answer, which is precisely the regime an RLVR reward relies on. The strong commercial judge is an explicit negative control: its 0.0% flip rate on both strata is based on 18 correct-candidate and 11 wrong-candidate prompts, so the zero-height bars indicate stable verdicts rather than missing observations.

To rule out the parsing explanation more directly, Figure 4 plots each judge's parse-success rate against its unstable-flip rate. The two are uncorrelated across judges: Qwen2.5-7B has near-perfect parsing yet the highest instability, so a well-formed verdict says nothing about whether that verdict is reproducible.

This comparison separates coverage from reliability. A parser failure reduces the number of usable rewards, but a parsed verdict that changes across repeated samples directly corrupts the reward label. The near-perfect parsing of Qwen2.5-7B together with its highest flip rate therefore rules out formatting as the main explanation. The same instability on correct and wrong candidates also means that it cannot be handled by filtering only difficult negatives; it is a property of the verifier signal itself.

4.3 How many samples are needed to trust a verdict?

If a verdict were driven by true correctness, a single judge call would suffice. We measure how often a majority vote over k samples agrees with the full (k=8) majority; Table 3 and Figure 6 report the

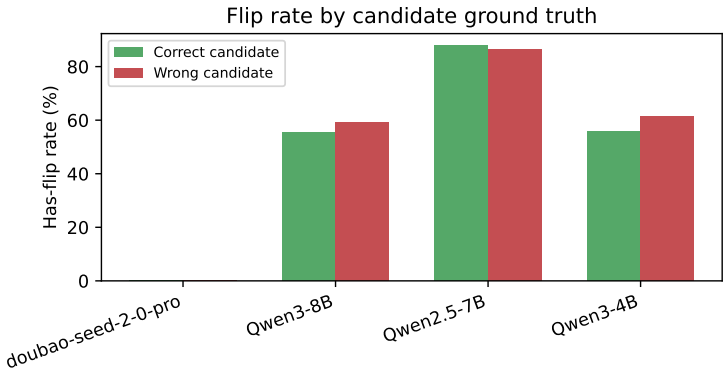

Figure 3: Has-flip rate by candidate ground truth. Instability is present on both correct and wrong candidates for every weak judge, confirming it is generic verdict noise rather than calibrated uncertainty on hard negatives.

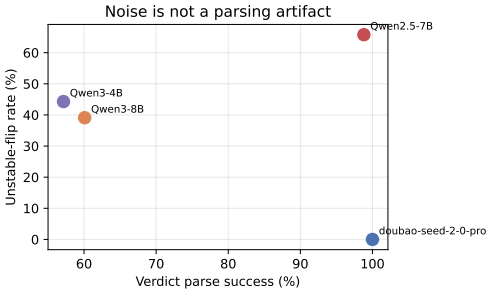

Figure 4: Parse-success rate vs. unstable-flip rate. High parse success does not imply stable verdicts: the judge that parses best (Qwen2.5-7B) is also the most unstable.

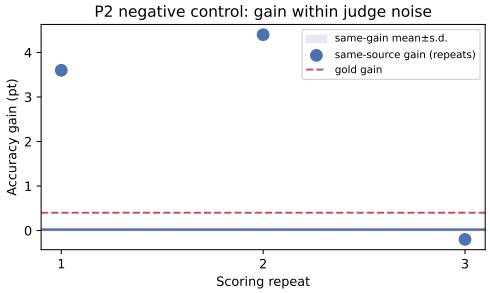

Figure 5: P2 negative control. The three repeated same-source gains (blue points) scatter across the mean±s.d. band (shaded) and span zero; the gold gain (red dashed) is negligible. The apparent single-run $+2.2$-point gain is an upper-tail draw of the judge's own scoring noise.

result. For the weak judges, a *single* call (the standard practice when using a judge as an RLVR reward) agrees with the well-sampled verdict only 75–87% of the time, meaning roughly one in five to one in four single-pass verdicts is unreliable; the strong judge is at 100% from $k{=}1$. This directly quantifies the cost of trusting a single weak-judge pass.

This curve also exposes a practical diminishing-returns regime. Moving from one sample to three recovers a substantial part of the lost agreement for every weak judge, while the gains from three to five samples are smaller and the final step to eight samples mainly provides a reference standard. This pattern suggests that the appropriate sampling budget should be selected from a pilot stability curve rather than fixed in advance. In particular, a judge with a 75–87% single-pass agreement rate should not be treated as equivalent to one with a higher agreement rate, even if both are nominally the same model class.

The diagnostic should report both the single-pass baseline and the smallest sample count that reaches a pre-specified agreement target, so that reward quality and serving cost are visible together.

4.4 Training-side negative control

We finally ask whether this judge noise couples to a trained policy. We run RLVR (GRPO) on a Qwen3-8B policy for MuSiQue answering, using a fixed same-source judge as the reward, and evaluate the base and trained policies with (i) the same-source judge and (ii) a gold exact/alias

Table 3: Agreement (%) between a k-sample majority vote and the full 8-sample majority, over prompts with all 8 verdicts parsed. A single weak-judge call disagrees with the well-sampled verdict on 13–25% of prompts; the strong judge is stable from k=1.

Judge	k=1	k=3	k=5	k=8
Qwen3-8B base (n=193)	87.2	93.0	96.2	100
Qwen2.5-7B base (n=867)	75.0	83.7	89.4	100
Qwen3-4B base (n=157)	82.6	89.1	92.9	100
doubao-seed-2-0-pro (n=26)	100	100	100	100

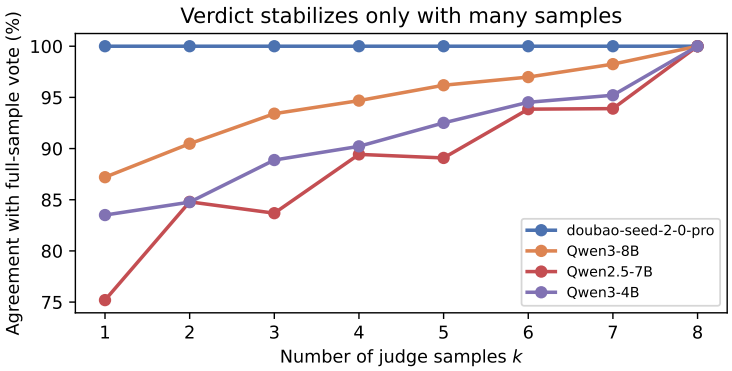

Figure 6: Agreement between a k-sample majority vote and the full-sample vote. Weak judges require many samples to stabilize, whereas a single strong-judge call already matches the well-sampled verdict.

matcher on a held-out set of 500 items. A single scoring pass suggested a positive same-source gain (+2.2 points), which would naively indicate the policy learned to please its own judge. However, because the judge is self-inconsistent (Section 4.1), we repeated the same-source scoring three times; Table 4 and Figure 5 show the gain is not robust.

The mean same-source gain (+2.6 points) is of the same magnitude as the judge's own scoring noise (s.d. 2.0 points; even the base policy's same-source accuracy drifts by ±2 points across passes), and the McNemar test on any single pass is far from significant. The gold gain is negligible. We conclude there is no robust trainable same-source gain here, which *also* reinforces the main message: because a single judge pass can flip roughly one in five verdicts (Table 3), any evaluation or reward that trusts a single weak-judge pass is unreliable.

4.5 Implications for RLVR practice

Taken together, the four experiments trace a single causal prerequisite. The weak open-source judges practitioners can afford to serve are self-inconsistent on a large fraction of prompts (Table 1), this inconsistency is generic verdict noise rather than calibrated uncertainty on hard cases (Table 2, Figure 3), it is not explained by parsing failures (Figure 4), and it is large enough that a single judge call—the default when a judge is used as an RLVR reward—disagrees with the well-sampled verdict on up to one in four prompts (Table 3, Figure 6). Each of these is a necessary condition for judge noise to corrupt the per-prompt reward statistics that GRPO-style allocation relies on.

Two practical consequences follow. First, *any* allocation heuristic that reads per-prompt reward variance as difficulty will, on these judges, spend a substantial share of its budget on prompts that are merely judge-unstable; the mean per-prompt variance we measure (0.11–0.17) is a direct estimate of the size of that misdirected component. Second, and independently, an evaluation that certifies a training gain from a single same-source judge pass is unsafe: our negative control shows an apparent +2.2-point gain that dissolves into the judge's own scoring noise once the evaluation is repeated

Table 4: Training-side negative control (n=500). The same-source gain from RLVR is within the judge's own scoring noise (three repeats span zero; McNemar on one pass p=0.77), and the gold gain is negligible. We therefore report this as a negative control: single-run judge scores cannot certify a training gain.

Metric	Value	Note
Gold gain (base → RLVR)	$+0.4$ pt	$0.082 \rightarrow 0.086$, negligible
Same-source gain, single run	$+2.2$ pt	would (mis)read as positive
Same-source gain, 3 repeats	$+3.6/+4.4/-0.2$ pt	mean $+2.6$, s.d. 2.0, spans 0
McNemar (one pass)	p=0.77	52 vs. 56 discordant

(Table 4, Figure 5). The remedy suggested by Table 3 is concrete—aggregating a handful of judge samples per verdict recovers most of the stability of a strong judge—but it multiplies the judge cost, which is precisely the cost practitioners were trying to avoid by choosing a weak open-source verifier. We therefore recommend running the cheap diagnostic of Section 3 before adopting an open-source judge as a reward, and budgeting for multi-sample verdicts when the measured instability is high.

5 Conclusion and Limitations

We presented a cheap, controlled diagnostic for judge self-inconsistency and showed that the weak open-source judges actually used as RLVR rewards in verify-hard domains are severely self-inconsistent—flipping their verdict on 39–66% of prompts—while a strong commercial judge is stable on the same inputs. This instability is not a parsing artifact, and its large per-prompt variance is exactly what a variance-driven rollout allocator would misread as difficulty. A training-side negative control further shows that a single weak-judge scoring pass cannot even certify a training gain, since the gain sits inside the judge's own scoring noise. We recommend that a judge-noise diagnostic be run before an open-source verifier is adopted as an RLVR reward.

Limitations. Our results establish the prerequisite for reward mis-allocation—large, prompt-correlated weak-judge variance—but not its causal effect on final accuracy in an end-to-end variance-driven training study; the training result is a negative control. Measurements use one verify-hard domain (MuSiQue), so magnitudes may differ for domains such as agent trajectory auditing. The unstable-flip threshold is a design choice, addressed by reporting full variance and has-flip statistics, while the smaller strong-judge control is only an adverse stability reference because of API-budget limits.

Ethics Statement

This work studies the reliability of automatic LLM judges used as rewards in reinforcement learning. All data are derived from MuSiQue, a publicly released multi-hop question-answering benchmark used under its original license; we introduce no new human-subject data collection and no personally identifiable information. Our central finding is a cautionary one: weak open-source judges are highly self-inconsistent, and single-pass judge scores should not be trusted as a sole reward or evaluation signal for high-stakes decisions without additional controls. We see no additional ethical risks beyond those common to automatic evaluation research, and we release our diagnostic so that practitioners can audit their own judges before deployment.

Reproducibility Statement

We designed the study for full reproducibility. Section 3 defines the judge-noise diagnostic (per-prompt Bernoulli variance, has-flip rate, and unstable-flip rate), the sampling protocol (k=8, τ=0.7), and the training-side negative-control setup. Appendix A details the MuSiQue verification suite and the exact evaluation splits; Appendix B lists the judge-serving configuration, the GRPO hyperparameters for the negative control, and all decoding settings. Every number in Table 1 and Table 4 is regenerated by the released scripts from the raw per-sample result files, and the repeated-scoring procedure behind Table 4 is included so the noise bands can be reproduced.

Use of Large Language Models

Large language models were used in two roles. (i) As *objects of study*: the judges (doubao-seed-2-0-pro, Qwen3-8B, Qwen2.5-7B, Qwen3-4B) and the RLVR policy (Qwen3-8B) evaluated in Section 4 are LLMs. (ii) As a *writing aid*: an LLM-based assistant was used for light editing (grammar, wording, and LaTeX formatting) of author-written text. All research ideas, experimental design, analysis, and claims are the authors' own; the authors verified every reported number against the raw result files and take full responsibility for the content.

References

Renat Aksitov, Sobhan Miryoosefi, Zonglin Li, Daliang Li, Sheila Babayan, Kavya Kopparapu, Zachary Fisher, Ruiqi Guo, Sushant Prakash, Pranesh Srinivasan, Manzil Zaheer, Felix Yu, and Sanjiv Kumar. Rest meets react: Self-improvement for multi-step reasoning llm agent, 2023. URL https://arxiv.org/abs/2312.10003.

Chris Alberti, Kenton Lee, and Michael Collins. A bert baseline for the natural questions, 2019. URL https://arxiv.org/abs/1901.08634.

Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety, 2016. URL https://arxiv.org/abs/1606.06565.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

Jacob Eisenstein, Chirag Nagpal, Alekh Agarwal, Ahmad Beirami, Alex D'Amour, DJ Dvijotham, Adam Fisch, Katherine Heller, Stephen Pfohl, Deepak Ramachandran, Peter Shaw, and Jonathan Berant. Helping or herding? reward model ensembles mitigate but do not eliminate reward hacking, 2024. URL https://arxiv.org/abs/2312.09244.

Jiazhan Feng, Shijue Huang, Xingwei Qu, Ge Zhang, Yujia Qin, Baoquan Zhong, Chengquan Jiang, Jinxin Chi, and Wanjun Zhong. Retool: Reinforcement learning for strategic tool use in llms, 2025. URL https://arxiv.org/abs/2504.11536.

Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization, 2022. URL https://arxiv.org/abs/2210.10760.

Veronika Hackl, Alexandra Elena Müller, Michael Granitzer, and Maximilian Sailer. Is gpt-4 a reliable rater? evaluating consistency in gpt-4 text ratings, 2023. URL https://arxiv.org/abs/2308.02575.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps, 2020. URL https://arxiv.org/abs/2011.01060.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Yejin Bang, Delong Chen, Wenliang Dai, Ho Shu Chan, Andrea Madotto, and Pascale Fung. Survey of hallucination in natural language generation, 2024. URL https://arxiv.org/abs/2202.03629.

Yuhua Jiang, Yuwen Xiong, Yufeng Yuan, Chao Xin, Wenyuan Xu, Yu Yue, Qianchuan Zhao, and Lin Yan. Pag: Multi-turn reinforced llm self-correction with policy as generative verifier, 2025. URL https://arxiv.org/abs/2506.10406.

Seungone Kim, Jamin Shin, Yejin Cho, Joel Jang, Shayne Longpre, Hwaran Lee, Sangdoo Yun, Seongjin Shin, Sungdong Kim, James Thorne, and Minjoon Seo. Prometheus: Inducing fine-grained evaluation capability in language models, 2024. URL https://arxiv.org/abs/2310.08491.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. Tulu 3: Pushing frontiers in open language model post-training, 2025. URL https://arxiv.org/abs/2411.15124.

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, 2025. URL https://arxiv.org/abs/2411.16594.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step, 2023. URL https://arxiv.org/abs/2305.20050.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. G-eval: Nlg evaluation using gpt-4 with better human alignment, 2023. URL https://arxiv.org/abs/2303.16634.

Ang Lv, Ruobing Xie, Xingwu Sun, Zhanhui Kang, and Rui Yan. The climb carves wisdom deeper than the summit: On the noisy rewards in learning to reason, 2025. URL https://arxiv.org/abs/2505.22653.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback, 2022. URL https://arxiv.org/abs/2203.02155.

Arjun Panickssery, Samuel R. Bowman, and Shi Feng. Llm evaluators recognize and favor their own generations, 2024. URL https://arxiv.org/abs/2404.13076.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model, 2024. URL https://arxiv.org/abs/2305.18290.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text, 2016. URL https://arxiv.org/abs/1606.05250.

Keita Saito, Akifumi Wachi, Koki Wataoka, and Youhei Akimoto. Verbosity bias in preference labeling by large language models, 2023. URL https://arxiv.org/abs/2310.10076.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

Rulin Shao, Shuyue Stella Li, Rui Xin, Scott Geng, Yiping Wang, Sewoong Oh, Simon Shaolei Du, Nathan Lambert, Sewon Min, Ranjay Krishna, Yulia Tsvetkov, Hannaneh Hajishirzi, Pang Wei Koh, and Luke Zettlemoyer. Spurious rewards: Rethinking training signals in rlvr, 2026. URL https://arxiv.org/abs/2506.10947.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Joar Skalse, Nikolaus H. R. Howe, Dmitrii Krasheninnikov, and David Krueger. Defining and characterizing reward hacking, 2025. URL https://arxiv.org/abs/2209.13085.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters, 2024. URL https://arxiv.org/abs/2408.03314.

Nisan Stiennon, Long Ouyang, Jeff Wu, Daniel M. Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul Christiano. Learning to summarize from human feedback, 2022. URL https://arxiv.org/abs/2009.01325.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Musique: Multi-hop questions via single-hop question composition, 2022. URL https://arxiv.org/abs/2108.00573.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. Solving math word problems with process- and outcome-based feedback, 2022. URL https://arxiv.org/abs/2211.14275.

Peiyi Wang, Lei Li, Liang Chen, Zefan Cai, Dawei Zhu, Binghuai Lin, Yunbo Cao, Qi Liu, Tianyu Liu, and Zhifang Sui. Large language models are not fair evaluators, 2023a. URL https://arxiv.org/abs/2305.17926.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models, 2023b. URL https://arxiv.org/abs/2203.11171.

Yidong Wang, Zhuohao Yu, Zhengran Zeng, Linyi Yang, Cunxiang Wang, Hao Chen, Chaoya Jiang, Rui Xie, Jindong Wang, Xing Xie, Wei Ye, Shikun Zhang, and Yue Zhang. Pandalm: An automatic evaluation benchmark for llm instruction tuning optimization, 2024. URL https://arxiv.org/abs/2306.05087.

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

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering, 2018. URL https://arxiv.org/abs/1809.09600.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason Weston. Self-rewarding language models, 2025. URL https://arxiv.org/abs/2401.10020.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena, 2023. URL https://arxiv.org/abs/2306.05685.

A Data and Evaluation Splits

Our verification suite is built from MuSiQue (Trivedi et al., 2022), a multi-hop question-answering benchmark. Each item is a (question, candidate answer, gold reference) triple, where the gold reference includes aliases used for exact/alias matching. For the judge-noise diagnostic (Section 4.1) we use the development split and sample each judge $k{=}8$ times per prompt at temperature $\tau{=}0.7$; the effective prompt counts per judge in Table 1 reflect items with at least two parsed verdicts (a prompt with fewer than two parsed samples is excluded from the variance statistics). The prompt-count differences across judges are driven by per-judge parsing success (the "Parsed" column), not by different underlying question sets.

For the training-side negative control (Section 4.4) we use a held-out policy-evaluation split of $n{=}500$ multi-hop questions. The base and RLVR policies each produce a single greedy answer per question, and both answer sets are scored under (i) the same-source judge and (ii) gold exact/alias matching. To characterise judge-scoring noise, the same-source scoring is repeated three times with independent judge sampling; Table 4 reports the resulting distribution of gains.

This appendix accompanies the released per-sample result files so that every statistic in the main tables can be recomputed: the raw judge samples underlying Table 1 and the per-item policy answers and repeated same-source scores underlying Table 4 are all included in the supplement.

B Judge Serving and Training Configuration

Judge sampling. Each judge is prompted with a fixed fact-checking system message that requests brief reasoning followed by a final line VERDICT: CORRECT or VERDICT: WRONG; we parse the final verdict and treat malformed outputs as *unparsed*. Sampling uses $k{=}8$, $\tau{=}0.7$ for the diagnostic. The strong commercial judge is queried through a hosted API under a fixed request-rate cap, which is why its prompt set in Table 1 is smaller; it is included only as an adverse stability control.

Negative-control training. The RLVR policy is Qwen3-8B trained with GRPO for the multi-hop answering task, using a fixed same-source judge served as an OpenAI-compatible reward endpoint. Reasoning-mode generation is disabled for both the policy and the judge so that answers are answer-first and short, which avoids truncation of the parsed answer. The policy is decoded greedily at evaluation time (one answer per question). All judge-serving flags, GRPO hyperparameters, and decoding settings are included as scripts in the supplement.