DIAGNOSE, THEN SELECT: WHAT MAKES CRITIQUE-BASED REFINEMENT EFFECTIVE?

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models increasingly rely on critique to refine their reasoning, yet existing approaches treat critique as an undifferentiated block of feedback and provide more information than may be necessary. We argue that critique-based refinement has two separate bottlenecks: the feedback must carry the causal information needed to repair an error, and the resulting revision must be reliably accepted or rejected. First, we investigate which parts of a critique make an error repairable by decomposing one structured record into isolated correctness-verdict, error-location, and error-diagnosis views, and compare them with the full critique. Across three refinement protocols, two model families, and two mathematical-reasoning benchmarks, diagnosis-only feedback recovers 98.8% of the wrong-to-correct rate of a full critique while using 30.9% of its feedback tokens; verdicts and locations recover only 42.6% and 62.3%. Correction alone, however, is an incomplete objective: critique also destabilizes answers that were already correct. Under solver–critic refinement on MATH-500, diagnosis corrects 8.8% of the weaker Llama solver's wrong answers but corrupts 25.7% of its correct ones, a net loss of 7.4 points, while the same feedback gains 3.8 points for the stronger Qwen solver. To address the second bottleneck, we propose DISEL-Consensus, a sampling-based revision selector that accepts a diagnosis-conditioned revision only when one independently sampled auxiliary revision supports it. At a common 1% calibration risk budget, Consensus reduces correct-to-wrong transitions from 13.6% under unconditional revision to 0.34%, retains 10.4% wrong-to-correct transitions, and gains 2.80 accuracy points with 923 marginal tokens per trial. Compared with DISEL-Pointwise selection baseline, DISEL-Consensus achieves a significantly lower corruption rate, while the difference in net accuracy is not statistically significant.

1 INTRODUCTION

Critique is widely used to improve large language model (LLM) reasoning. Self-refinement and reflection generate natural-language feedback and revise an initial response (Madaan et al., 2023; Shinn et al., 2023), while related work uses critiques as supervision or to assist human oversight (Wang et al., 2025; Xie et al., 2025; Zhang et al., 2026; Saunders et al., 2022; McAleese et al., 2024). Yet refinement based on critique is not reliably beneficial. Models may fail to identify their errors, and unconstrained self-correction can reduce accuracy (Huang et al., 2024; Kamoi et al., 2024). Existing work largely asks whether a critique is correct or where reasoning first fails (Tyen et al., 2024; Samanta et al., 2026). We instead ask: *which information in a critique makes an error repairable?* A full critique may state that an answer is wrong, locate the error, explain why it fails, and suggest a fix; more detail can help, but it costs tokens and may distract the reviser. We generate one structured critique with separately labeled verdict, error-location, and diagnosis fields, and compare field-restricted views of that record with the full critique under solver–critic refinement (Saunders et al., 2022), self-refinement (Madaan et al., 2023), and two-model debate (Du et al., 2024). The primary solver–critic setting pairs each solver with a stronger critic, so that differences between views reflect what the critique says rather than whether the critic found the error (Tyen et al., 2024; Huang et al., 2024); the two same-scale protocols test whether the conclusions hold when the feedback source is no stronger than the solver.

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.

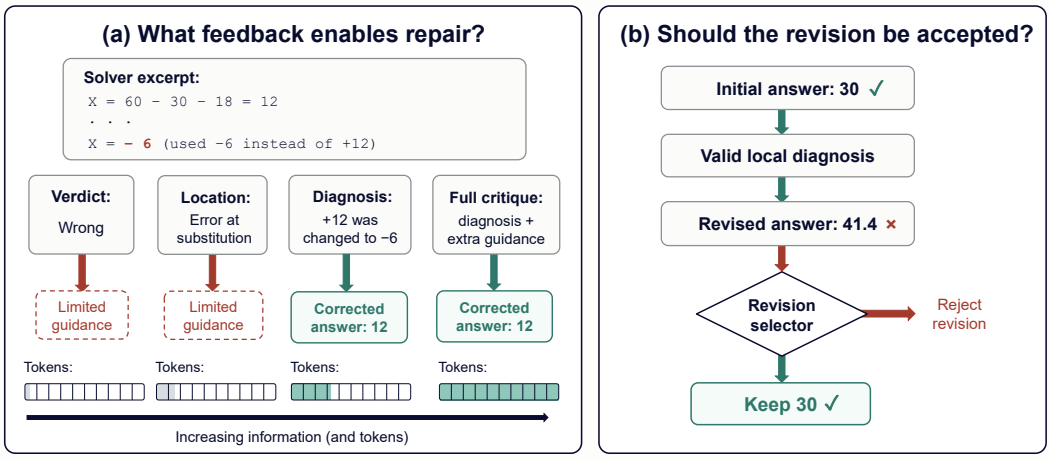


Figure 1: **Critique-based refinement has two bottlenecks.** **(a)** feedback must make an error repairable; diagnosis exposes the causal inconsistency, while a full critique adds guidance and tokens. **(b)** a valid diagnosis can still induce a harmful revision, motivating explicit selection between the initial and revised answers.

Critique-based refinement can fail at two stages (Figure 1). Feedback may identify an error without providing enough causal information to repair it, and even valid feedback may induce a revision worse than the original answer. The left example (Figure 1) first establishes $x = 60 - 30 - 18 = 12$ but later substitutes $x = -6$. A verdict or location flags the failure. Only the diagnosis states that the sign change is unjustified and enables repair. The right example shows the complementary risk: a correct answer of 30 becomes 41.4 after a locally valid diagnosis. Thus local feedback quality does not guarantee a globally better trajectory.

Because a revision can go either way, counting the errors it repairs overstates what refinement achieves. We therefore measure both the wrong-to-correct rate ($w \to c$), how often revision repairs an initially wrong answer, and the correct-to-wrong rate ($c \to w$), how often it corrupts an initially correct one. The two can diverge sharply: under solver–critic refinement on MATH-500, diagnosis repairs 8.8% of the weaker Llama solver's wrong answers but corrupts 25.7% of its correct ones, whereas Qwen loses only about 1% of its correct answers. These aggregate rates show when accepting every revision is unsafe, but not which individual revision to reject. We therefore add revision selection as a separate step.

Three questions organize the paper. *Q1. Is a full critique necessary?* We hypothesize that its correction benefit is concentrated in a subset of its components, so most feedback tokens are dispensable. *Q2. Is better feedback sufficient?* We hypothesize that it is not, because the same feedback that repairs errors can also corrupt correct answers, and for some models the corruption dominates. *Q3. Can revision selection resolve this problem?* We hypothesize that selection evidence can retain genuine corrections while preventing critique-induced corruptions. Based on these questions, we make three contributions as below:

- Diagnosis recovers essentially all full-critique correction benefit at about one third of the feedback tokens, while verdict and location are often insufficient (§ 2).
- We measure correction jointly with the corruption it induces and show that informative feedback can still lower accuracy: the same diagnosis that raises Qwen's accuracy by 3.8 points on MATH-500 lowers the weaker Llama solver's by 7.4 points, and the sign of the effect is the same under all three protocols. Revision acceptance is therefore a decision problem distinct from critique quality (§2).
- We introduce DISEL-Consensus, a sampling-based revision selector that uses one auxiliary revision. At a common 1% calibration risk budget it reduces solver–critic corruption from 13.6% to 0.34% while retaining 10.4% correction, and we report its full correction–corruption and token-cost trade-off (§ 3–4).

Verdict (v)

OBJECT.

Error Location (ℓ)

$500 = 5^2 \times 2^2$.

Diagnosis (d)

500 equals $2^2 \times 5^3$, not $2^2 \times 5^2$. The exponent on 5 is incorrect, leading to a flawed divisor sum calculation.

Full Critique

Verdict + Error Location + Diagnosis +

Corrective Hint

Factor 500 as $2^2 \times 5^3$ and apply the divisor sum formula with corrected exponents.

Alternative Approach

Compute A using $\sigma(500) = \sigma(2^2) \times \sigma(5^3)$, factor the resulting A, then identify and sum its distinct prime factors.

Figure 2: **One structured critique record $c = (v, \ell, d)$ and the full critique.** In a divisor-sum problem, the initial solution factorizes 500 as $2^2 \times 5^2$. The verdict only objects to the answer, the error location points to the faulty factorization, and only the diagnosis states the correct exponent needed to repair it. The full critique contains all three fields plus a corrective hint and an alternative approach.

2 WHAT MAKES CRITIQUE USEFUL?

This section takes up *Q1* and *Q2*. Both ask what a critique must *contain*, so we hold the underlying critique record fixed and vary only what the reviser is shown. Each view is measured by the correction it retains and the feedback tokens it requires, which answers *Q1*, and by how often its revision overturns an answer that was already correct, which answers *Q2*.

Critique decomposition. We decompose critique along three levels of information already distinguished by neighboring verification paradigms (Figure 2). A *verdict* corresponds to outcome-level supervision. It says whether the response is acceptable but not where its reasoning fails (Cobbe et al., 2021). An *error location* adds process-level localization, motivated by work showing that models can often repair an error when its position is supplied (Tyen et al., 2024; Uesato et al., 2022; Lightman et al., 2024). A *diagnosis* adds a causal account of why the identified step is invalid and how that failure propagates. This last distinction is the target of our study. Localization can identify the step to revisit without supplying the mathematical relation needed to change it correctly. Structured error-localization work further motivates treating location as an explicit field rather than inferring it from free-form feedback (Samanta et al., 2026).

Given a question q, language model π, and initial solution $y_0 \sim \pi(\cdot \mid q)$, a critic produces one structured record $c = (v, \ell, d)$ containing a verdict v, error location ℓ, and diagnosis d. We deterministically render four views from this same record: **verdict field only** (v), **error-location field only** (ℓ), **diagnosis field only** (d), or the **full critique** (v, ℓ, d). Consequently, differences between conditions reflect the information exposed to the reviser rather than variation from independently sampled critiques. The fields are operational prompt components, not claims of semantic independence: diagnosis can reveal both the verdict and the location implicitly.

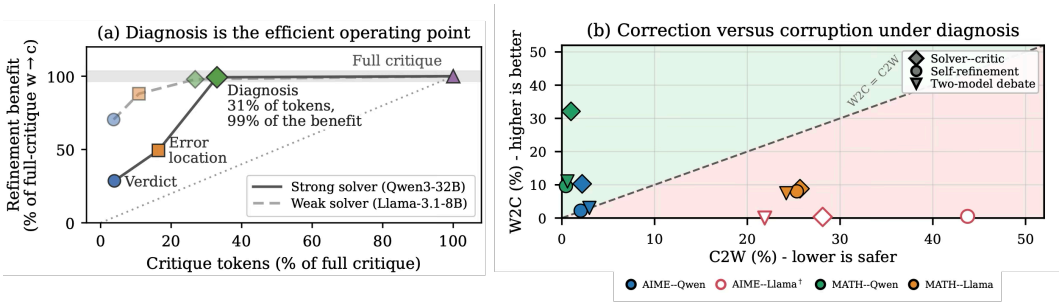


Figure 3: **(a)** Correction and feedback tokens normalized by full critique within each cell (mean ± standard error; nine non-degenerate cells). **(b)** Diagnosis-only wrong-to-correct (W2C) versus correct-to-wrong (C2W) rates for the 12 benchmark–solver–protocol cells. Points toward the upper left combine greater correction with less corruption, while the dashed diagonal marks equal conditional transition rates. Marker shape denotes the refinement protocol and color denotes the benchmark–solver pair. Because W2C and C2W condition on different initial-answer cohorts, position relative to the diagonal alone does not determine the net accuracy effect. Hollow AIME–Llama points are exploratory because the initially correct cohort contains only two histories.

Figure 2 gives a compact example in which A is the sum of the divisors of 500 and the problem asks for the sum of A's distinct prime factors. The initial response uses the faulty factorization $500 = 2^2 \times 5^2$. The component views successively reveal that an error occurred, where it occurred, and why the exponent is wrong.

Setup. We evaluate two mathematical-reasoning benchmarks, MATH-500 (Hendrycks et al., 2021) and all 180 AIME problems from 2020–2025, with two model families—Qwen3-32B critiqued by Qwen3-235B-A22B (Yang et al., 2025) and Llama-3.1-8B critiqued by Llama-3.1-70B (Grattafiori et al., 2024)—under three interaction protocols: solver–critic (Saunders et al., 2022), self-refinement (Madaan et al., 2023), and two-model debate (Du et al., 2024). The three AIME–Llama cells are degenerate, since the full critique there produces $w \to c < 0.5\%$ and leaves the normalized component ratios undefined; we report them descriptively and compute all averages over the remaining nine. We summarize the design here and defer full details to §4.1.

Each record is generated once, by the paired critic, the solver itself, or a same-size peer depending on the protocol, and all four views are rendered from it. Revision calls differ only in the rendered view: the problem, initial solution, revision instruction, and decoding configuration are held fixed, and omitted fields are not exposed elsewhere in the prompt. The ablation is therefore a prompt-level intervention on naturally generated critique content. Isolation is at the level of fields rather than information, since a diagnosis typically refers to the step it explains.

Observation 1: Diagnosis is the efficient operating point. Figure 3(a) answers *Q1*. Diagnosis recovers nearly all of the correction benefit of a full critique while using approximately one third of its feedback tokens. Verdict and error location are shorter, but neither matches the correction performance of diagnosis. The benefit is therefore concentrated in the causal explanation: moving from location to diagnosis provides the largest improvement in correction relative to its additional token cost. This sharpens a prior claim. Tyen et al. (2024) report that models can correct reasoning errors once given the error location, which suggests that localization is the scarce ingredient; in our setting it recovers barely two-thirds of what the explanation does. This qualifies rather than contradicts Tyen et al. (2024). They supply ground-truth locations and regenerate from the identified step, whereas we pass a critic's localization as text to an independent reviser.

Observation 2: Diagnosis alone is not sufficient. Turning to Q2, Figure 3(b) evaluates both the corrections and corruptions produced by diagnosis-conditioned revision. For the smaller Llama solver on MATH-500, initial accuracy exceeds the break-even accuracy, so always accepting the revision lowers overall accuracy. The stronger Qwen solver remains below its break-even point and benefits from revision. Better feedback is therefore not sufficient: its net effect depends jointly on the solver's initial accuracy and its $w \to c$ and $c \to w$ rates. Diagnosis can be the right *content* while accepting the resulting revision is still the wrong *decision*. This motivates DISEL: use diagnosis as the compact

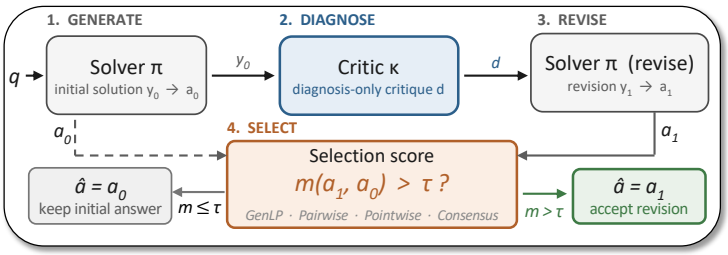


Figure 4: **DiSel (Diagnose-then-Select).** A diagnosis conditions a primary revision, while one independently sampled auxiliary revision provides a second opinion. A revision selector then decides whether the primary revision should replace the initial answer.

repair signal, then separately decide whether its revision should replace the initial answer. Whether that decision can in fact be made reliably is Q3, which Section 4 evaluates.

3 DiSel: Diagnose, then Select

Section 2 shows that diagnosis is an effective and token-efficient repair signal, but also that automatically accepting every diagnosis-induced revision can corrupt answers that were already correct. We therefore separate refinement into two steps: *diagnose* the possible error, then *select* whether the resulting revision should replace the initial answer as shown in Figure 4.

Revision-selection view. The second step selects between two fixed candidates rather than generating another correction. Let e_m denote selector-specific evidence (generation or verifier scores for the learned selectors and the auxiliary revision y_2 for Consensus) and let $s_m(q, y_0, y_1, d, e_m)$ denote the score assigned by selector m to replacing the initial solution with the primary revision. Given a selector-specific threshold τ_m, the final solution is

$$\hat{y} = \begin{cases} y_1, & s_m(q, y_0, y_1, d) > \tau_m, \\ y_0, & \text{otherwise.} \end{cases} \tag{1}$$

Each revision selector below supplies a different score s_m, while the candidate pair, bypass rules, threshold-calibration objective, and decision rule remain fixed. The formulation therefore isolates the evidence used to accept a proposed revision: it asks whether y_1 should replace y_0, rather than asking either model to generate another candidate or requiring an absolute correctness judgment.

Diagnose, then compare. Given a question q, a solver first produces an initial solution y_0 with answer a_0. A critic supplies a diagnosis d. The revision prompt concatenates q, the frozen y_0, and d, and asks the solver to return a corrected solution. This produces a primary revision y_1 with answer a_1. If $a_1 \equiv a_0$, no selection is needed and the answer remains unchanged. If the answers differ, DiSel-Consensus issues the same revision prompt once more with an independent decoding sample, producing an auxiliary revision y_2 and answer a_2. The auxiliary answer acts as an independent vote:

Auxiliary outcome	Interpretation	Basic decision ($\tau = 0$)
$a_2 \equiv a_1$	Supports the revision	Use a_1
$a_2 \equiv a_0$	Supports the initial answer	Keep a_0
Neither answer	Agreement is unresolved	Keep a_0

The table shows the basic $\tau = 0$ rule. Out-of-group calibration may choose a stricter or more permissive threshold.

Here, $\equiv$ denotes answer equivalence under the benchmark grader. Unchanged or unparseable primary revisions bypass selection and retain a_0. Therefore, the basic mechanism is as follows: when two independently sampled, diagnosis-conditioned revisions reach the same answer, their agreement provides evidence for accepting that answer.

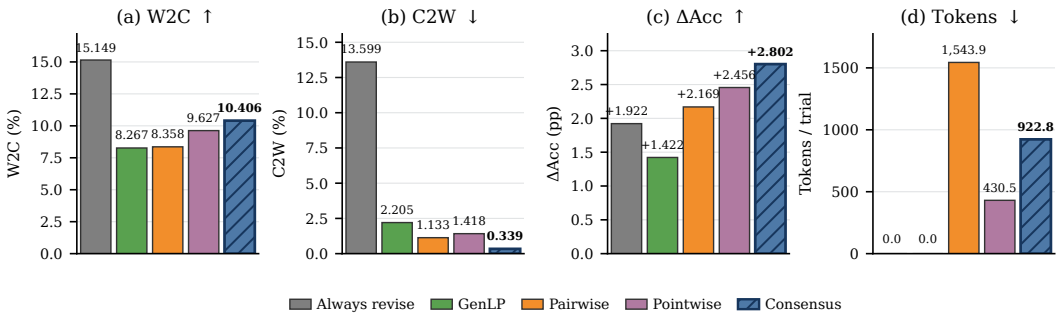


Figure 5: **Aggregate solver–critic revision-selection results at the common calibration budget $\rho = 1\%$.** Bars report **(a)** wrong-to-correct rate, **(b)** correct-to-wrong rate, **(c)** accuracy change, and **(d)** exact marginal prompt-plus-completion tokens per trial. Values are macro-averaged equally across six benchmark–model-pair slices. Higher is better in panels (a) and (c), whereas lower is better in panels (b) and (d).

Auxiliary pairing and deployment. Our frozen evaluation contains 16 independently sampled diagnosis-conditioned revisions for each initial trajectory. When repeat j is treated as the primary revision, we use the next repeat cyclically, $j^+ = 1 + (j \bmod 16)$, as its auxiliary revision. Thus, every repeat serves once as a primary revision and once as an auxiliary revision, and no revision is paired with itself. This pairing is fixed before evaluation and does not depend on correctness labels.

At deployment, the procedure is sequential. The solver first generates the primary revision. If its answer is unparseable or equivalent to the initial answer, DISEL retains the initial answer without requesting an auxiliary revision. Otherwise, the solver generates one additional revision from the same question, initial solution, and diagnosis using an independent decoding sample. The resulting auxiliary answer is then used by DISEL-Consensus to select between the initial and primary answers.

Revision selectors. The general selection problem is to decide whether a_1 should replace a_0. We compare four revision selectors that differ only in the evidence used for this decision:

- **DISEL-GenLP** compares the generation likelihoods of the two candidate answers.
- **DISEL-Pairwise** asks a verifier to compare the two candidates directly.
- **DISEL-Pointwise** asks a verifier to evaluate each candidate independently and compares their correctness scores.
- **DISEL-Consensus** uses agreement with the auxiliary revision a_2.

No verifier prompt contains the reference answer or correctness label. Each selector converts its evidence into a selection score, and the revision is accepted only when that score exceeds a selection threshold τ. For DISEL-Consensus, support for the revision, an unresolved vote, and support for the initial answer correspond to ordered scores of $+1$, 0, and -1, respectively. A stricter threshold accepts fewer revisions, while a more permissive threshold retains more potential corrections at the risk of accepting more harmful changes.

Section 4.1 describes how τ is selected on calibration data using the corruption budget ρ. The budget ρ is not part of the consensus mechanism itself. It specifies the desired correction–corruption operating point, while τ implements that choice for a particular selector.

4 EXPERIMENTS

This section completes the evidence for *Q1* and *Q2* and takes up *Q3*. To test whether revision acceptance can be decided reliably, we hold the initial answer, diagnosis, and resulting candidate revision fixed. The selectors differ only in the evidence used to choose between the initial and revised answers. We compare each selector with Always revise and report both the corrections it retains and the corruptions it prevents, so that a method is not rewarded for reducing corruption simply by

Table 1: **Revision-selection results by benchmark and model pair.** Model pairs are listed as solver/critic. All selectors operate on the same diagnosis-conditioned revisions and are calibrated out of group at the common $\rho = 1\%$ risk budget. ΔAcc is the percentage-point change from initial accuracy, and Tok. is the marginal prompt-plus-completion tokens per trial. Boldface identifies our proposed selector, not necessarily the best result in every slice. For AIME with Llama-8B/Llama-70B, the initially correct cohort contains only four histories, so its corruption and net-accuracy estimates are exploratory.

		AIME				**MATH-500**			
Model pair	Selector	$w \to c$	$c \to w$	ΔAcc	Tok.	$w \to c$	$c \to w$	ΔAcc	Tok.
Llama-70B/ Qwen-235B	Always revise	7.49	42.36	+0.53	0	32.12	4.93	+6.19	0
	DiSel-GenLP	0.62	8.33	−0.63	0	17.75	1.04	+4.60	0
	DiSel-Pairwise	2.59	1.04	+2.08	2,729	11.20	1.10	+2.59	775
	DiSel-Pointwise	1.80	1.04	+1.41	783	19.75	1.08	+5.17	240
	DiSel-Consensus	0.00	0.00	+0.00	1,724	27.17	0.43	+7.85	468
Llama-8B/ Llama-70B	Always revise	1.84	12.50	+0.96	0	14.22	18.62	−4.02	0
	DiSel-GenLP	0.31	0.00	+0.29	0	0.89	1.09	−0.21	0
	DiSel-Pairwise	0.00	1.56	−0.10	1,411	2.08	1.09	+0.32	836
	DiSel-Pointwise	0.21	3.13	+0.00	436	1.98	1.01	+0.32	273
	DiSel-Consensus	1.74	0.00	+1.64	936	0.00	0.00	+0.00	551
Qwen-32B/ Qwen-235B	Always revise	9.86	2.14	+6.10	0	25.36	1.05	+1.77	0
	DiSel-GenLP	4.67	1.81	+2.63	0	25.36	0.96	+1.85	0
	DiSel-Pairwise	9.64	1.15	+6.25	3,140	24.64	0.86	+1.86	373
	DiSel-Pointwise	9.26	1.48	+5.89	753	24.76	0.78	+1.95	99
	DiSel-Consensus	8.28	0.66	+5.48	1,657	25.24	0.95	+1.85	201

rejecting all revisions. We also report the marginal tokens required by each selector and calibrate selection thresholds out of group, ensuring that no selector is tuned on the evaluation cells on which it is scored.

4.1 Experimental Setup and Baselines

We evaluate MATH-500 (Hendrycks et al., 2021) and all 180 AIME problems from 2020–2025 using Qwen3-32B/Qwen3-235B-A22B (Yang et al., 2025) and Llama-3.1-8B/Llama-3.1-70B (Grattafiori et al., 2024); the revision-selection study additionally includes the cross-family Llama-3.1-70B/Qwen3-235B-A22B pair, yielding six benchmark–model-pair slices. Each frozen trajectory has 16 independently sampled diagnosis-conditioned revisions, with each repeat serving once as the primary revision and the next cyclic repeat as its DiSel-Consensus auxiliary. Solver–critic is the primary setting, with self-refinement and two-model debate as robustness checks. Generations use temperature 0.6 and top-$p = 0.95$, while verifier calls are deterministic with 64/128-token caps for AIME/MATH-500. The auxiliary uses the same prompt and decoding configuration as the primary revision. We exclude unparseable initial answers, count unparseable revisions as incorrect, and grade AIME by exact integer matching and MATH-500 by symbolic equivalence. Always revise accepts and is penalized for an unparseable revision, whereas every gated selector applies the shared parser bypass and retains y_0. Gains over Always revise therefore include this abstention benefit. The selector study contains 81,936 revision rows from 5,121 frozen histories.

The critique-component baselines are verdict, error location, diagnosis, and full critique, rendered from the same structured record within a fixed revision prompt. For revision selection, we compare DiSel-Consensus with Always revise, DiSel-GenLP, DiSel-Pairwise, and DiSel-Pointwise on identical frozen candidate pairs using the same answer parser and benchmark-specific equivalence test. We report $w \to c$, $c \to w$, accuracy change, selection rate, and exact marginal prompt-plus-completion tokens. Confidence intervals and paired tests use 10,000 correctness-stratified, problem-clustered bootstrap samples that keep all 16 repeats together. Thresholds are calibrated out of group—leave-one-year-out for AIME and leave-one-subject-out for MATH-500—by maximizing $w \to c$ subject to $c \to w \leq \rho$. We fix $\rho = 1\%$ for the main comparison and report the complete ρ sweep.

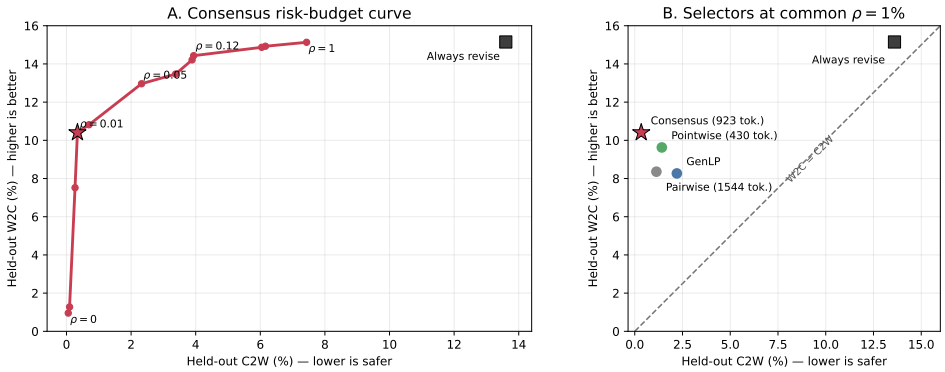


Figure 6: **Revision selection trades off correction and corruption. Left:** Held-out operating points of DISEL-Consensus as the calibration risk budget ρ varies. Increasing ρ makes the selector more permissive, retaining more corrections while admitting more corruptions. The star marks the prespecified $\rho = 1\%$ operating point, and the square marks Always revise. **Right:** Held-out operating points of all revision selectors at the common $\rho = 1\%$ calibration budget. Points toward the upper left represent a more favorable trade-off: higher $w \to c$ and lower $c \to w$. Values are macro-averaged equally across six benchmark–model-pair slices.

4.2 REVISION-SELECTION RESULTS

We compare all revision selectors on identical frozen candidate pairs. A selector is useful only when the corruptions it prevents outweigh the genuine corrections it rejects. We first report each benchmark–model-pair slice separately in Table 1. The equal-weight macro average is reported in Figure 5.

Consensus gives the most favorable aggregate operating point. Across the six benchmark–model-pair slices, DISEL-Consensus prevents nearly all of the corruptions caused by Always revise while retaining more than two thirds of its successful corrections, the largest macro-averaged accuracy gain among the evaluated selectors. It outperforms GenLP and Pairwise in net accuracy. Pointwise is the strongest competing baseline: Consensus produces fewer corruptions, but their difference in final accuracy is not statistically resolved, and neither is the advantage of Consensus over Always revise. Figure 6 places these results on a common axis. At the prespecified 1% budget, Consensus occupies the most favorable upper-left point, and relaxing ρ moves it upward and to the right as both corrections and corruptions increase. We retain $\rho = 1\%$ rather than tuning it on evaluation performance. We therefore claim a favorable aggregate correction–corruption operating point, not uniform statistical superiority.

The advantage is not uniform across slices or protocols. Consensus appears strong for AIME with the smaller Llama solver, although this slice is exploratory because its initially correct cohort contains only four histories, and performs particularly well for MATH-500 with the cross-family Llama-70B/Qwen-235B pair. Under the strict 1% budget, however, it abstains from all revisions in the AIME cross-family and MATH-500 Llama-8B/Llama-70B slices. DISEL-Pairwise has the largest gain for AIME with Qwen, and DISEL-Pointwise is strongest for MATH-500 with Qwen. The same pattern holds across protocols. DISEL-Consensus is strongest in the primary solver–critic setting and roughly tied with DISEL-Pointwise under two-model debate, while DISEL-Pointwise is better under self-refinement. DISEL-Consensus is thus a strong sampling-based selector in the primary setting rather than a universally best rule.

Selection cost. Consensus adds approximately 923 prompt-plus-completion tokens per trial, more than Pointwise and less than Pairwise (Figure 5). Pointwise is more token-efficient, yielding 1.24 additional accuracy points per 1,000 marginal tokens over Always revise, compared with 0.95 for Consensus. Consensus is therefore not a zero-cost selector, but it requires no log-probability-capable endpoint.

5 Related Work

Critique-based refinement. Self-Refine (Madaan et al., 2023) and Reflexion (Shinn et al., 2023) established generate–critique–revise loops. Later work trains critique generators (Wang et al., 2025; Xie et al., 2025; Zhang et al., 2026) or learned correctors (Welleck et al., 2023), grounds revision in execution, tools, or retrieved evidence (Chen et al., 2024b; Gou et al., 2024; Gao et al., 2023), and uses language or model-generated feedback as training signal (Scheurer et al., 2023; Bai et al., 2022). Critiques can also aid human oversight (Saunders et al., 2022; McAleese et al., 2024), but critics hallucinate flaws (McAleese et al., 2024), and intrinsic self-correction often degrades accuracy without a reliable external signal (Huang et al., 2024; Kamoi et al., 2024). We therefore measure $c \to w$ transitions alongside correction, rather than treating every critical intervention as benign.

Critique content and debate. Outcome verifiers (Cobbe et al., 2021), process supervision (Uesato et al., 2022; Lightman et al., 2024), and self-verification (Weng et al., 2023) establish the value of inspecting reasoning rather than only final answers, but do not isolate which natural-language information a reviser needs. Tyen et al. (2024) identify error location as a key signal, and Samanta et al. (2026) study structured localization. We instead hold one critique record fixed and expose its semantic fields separately, distinguishing knowing *where* an error occurs from knowing *why* it is invalid. We also test this distinction in two-model debate, originally a scalable-oversight protocol (Irving et al., 2018) and since used to diversify reasoning (Du et al., 2024; Liang et al., 2024), reach consensus (Chen et al., 2024a), build evaluators (Chan et al., 2024), and help weaker judges (Khan et al., 2024). Our condition is narrower: it treats the peer message as a structured critique under the same ablations.

Confidence, calibration, and selective generation. Classical calibration asks whether predicted confidence matches empirical correctness. Modern neural networks often require post-hoc calibration such as temperature scaling (Guo et al., 2017). Calibration has been studied for pretrained transformers (Desai & Durrett, 2020) and generative question answering (Jiang et al., 2021). For large language models, pointwise correctness estimation probes whether a model can estimate the correctness of its own proposed answer (Kadavath et al., 2022), while verbalized and black-box confidence elicitation provide alternatives when token probabilities are unavailable or poorly calibrated (Tian et al., 2023; Xiong et al., 2024). Pointwise and pairwise self-evaluation can support selective generation (Ren et al., 2023). Sampling-based methods instead measure disagreement across generations, either at the level of semantic equivalence classes (Kuhn et al., 2023) or factual consistency (Manakul et al., 2023). We use these signals for revision selection between a frozen initial answer and its diagnosis-conditioned revision, and evaluate them using both $w \to c$ corrections and $c \to w$ corruptions.

6 Conclusion

Critique-based refinement has two separable bottlenecks. Diagnosis is the efficient content unit: it retains nearly all full-critique correction performance at roughly one third of the feedback tokens, while verdict and location omit information that is often necessary for repair. Acceptance can be improved with a simple black-box signal. DiSel-Consensus uses one auxiliary revision to test semantic agreement. At the prespecified 1% calibration risk budget, it reduces solver–critic corruption from 13.6% to 0.34% while retaining 10.4% correction and producing the strongest observed accuracy gain. Its net-accuracy advantage over Pointwise is not statistically resolved, and it adds 923 marginal tokens per trial, so the contribution is a safer, portable operating point rather than a uniformly dominant selector. Our study is limited to mathematical reasoning and open-weight model families. Broader tasks, shorter auxiliaries, and external validation are important next steps.

Limitations. While our framework demonstrates clear token efficiency and safety trade-offs, this study focuses on mathematical reasoning benchmarks (MATH-500 and AIME) with deterministic verification. Evaluating whether the diagnostic efficiency and the DiSel-Consensus revision selector extend to open-ended generation, factual querying, or code generation remains an open question. Additionally, our empirical evaluation is centered on open-weight Qwen and Llama models and single-turn refinement. Investigating shorter auxiliaries, multi-turn critique dynamics, and larger proprietary reasoning models provides promising directions for future work.

AI USE STATEMENT

We used generative AI tools for surveying related work, polishing the writing and running experiments, not for research ideation, experimental design. We take full responsibility for the final content of this work, including any text, claims, or artifacts produced with the aid of generative AI.

REFERENCES

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, et al. Constitutional AI: Harmlessness from AI feedback. *arXiv preprint arXiv:2212.08073*, 2022.

Chi-Min Chan, Weize Chen, Yusheng Su, Jianxuan Yu, Wei Xue, Shanghang Zhang, Jie Fu, and Zhiyuan Liu. ChatEval: Towards better LLM-based evaluators through multi-agent debate. In *The Twelfth International Conference on Learning Representations*, 2024.

Justin Chen, Swarnadeep Saha, and Mohit Bansal. ReConcile: Round-table conference improves reasoning via consensus among diverse LLMs. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7066–7085, 2024a. doi: 10.18653/v1/2024.acl-long.381.

Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. Teaching large language models to self-debug. In *The Twelfth International Conference on Learning Representations*, 2024b.

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

Shrey Desai and Greg Durrett. Calibration of pre-trained transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 295–302, 2020. doi: 10.18653/v1/2020.emnlp-main.21.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 11733–11763. PMLR, 2024.

Luyu Gao, Zhuyun Dai, Panupong Pasupat, Anthony Chen, Arun Tejasvi Chaganty, Yicheng Fan, Vincent Zhao, Ni Lao, Hongrae Lee, Da-Cheng Juan, and Kelvin Guu. RARR: Researching and revising what language models say, using language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 16477–16508, 2023. doi: 10.18653/v1/2023.acl-long.910.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. CRITIC: Large language models can self-correct with tool-interactive critiquing. In *The Twelfth International Conference on Learning Representations*, 2024.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1321–1330. PMLR, 2017.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. In *The Twelfth International Conference on Learning Representations*, 2024.

Geoffrey Irving, Paul Christiano, and Dario Amodei. AI safety via debate. *arXiv preprint arXiv:1805.00899*, 2018.

Zhengbao Jiang, Jun Araki, Haibo Ding, and Graham Neubig. How can we know when language models know? on the calibration of language models for question answering. *Transactions of the Association for Computational Linguistics*, 9:962–977, 2021. doi: 10.1162/tacl_a_00407.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, et al. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*, 2022.

Ryo Kamoi, Yusen Zhang, Nan Zhang, Jiawei Han, and Rui Zhang. When can LLMs actually correct their own mistakes? a critical survey of self-correction of LLMs. *Transactions of the Association for Computational Linguistics*, 12:1417–1440, 2024. doi: 10.1162/tacl_a_00713.

Akbir Khan, John Hughes, Dan Valentine, Laura Ruis, Kshitij Sachan, Ansh Radhakrishnan, Edward Grefenstette, Samuel R. Bowman, Tim Rocktäschel, and Ethan Perez. Debating with more persuasive LLMs leads to more truthful answers. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 23662–23733. PMLR, 2024.

Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. In *The Eleventh International Conference on Learning Representations*, 2023.

Tian Liang, Zhiwei He, Wenxiang Jiao, Xing Wang, Yan Wang, Rui Wang, Yujiu Yang, Shuming Shi, and Zhaopeng Tu. Encouraging divergent thinking in large language models through multi-agent debate. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 17889–17904, 2024. doi: 10.18653/v1/2024.emnlp-main.992.

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. In *The Twelfth International Conference on Learning Representations*, 2024.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-Refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36: 46534–46594, 2023. doi: 10.52202/075280-2019.

Potsawee Manakul, Adian Liusie, and Mark Gales. SelfCheckGPT: Zero-resource black-box hallucination detection for generative large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 9004–9017, 2023. doi: 10.18653/v1/2023.emnlp-main.557.

Nat McAleese, Rai Michael Pokorny, Juan Felipe Ceron Uribe, Evgenia Nitishinskaya, Maja Trebacz, and Jan Leike. LLM critics help catch LLM bugs. *arXiv preprint arXiv:2407.00215*, 2024.

Jie Ren, Yao Zhao, Tu Vu, Peter J. Liu, and Balaji Lakshminarayanan. Self-evaluation improves selective generation in large language models. In *Proceedings on "I Can't Believe It's Not Better: Failure Modes in the Age of Foundation Models" at NeurIPS 2023 Workshops*, volume 239 of *Proceedings of Machine Learning Research*, pp. 49–64. PMLR, 2023.

Ankur Samanta, Akshayaa Magesh, Ayush Jain, Kavosh Asadi, Youliang Yu, Daniel Jiang, Boris Vidolov, Kaveh Hassani, Paul Sajda, Jalaj Bhandari, and Yonathan Efroni. Structure enables effective self-localization of errors in LLMs. *arXiv preprint arXiv:2602.02416*, 2026. doi: 10.48550/arXiv.2602.02416.

William Saunders, Catherine Yeh, Jeff Wu, Steven Bills, Long Ouyang, Jonathan Ward, and Jan Leike. Self-critiquing models for assisting human evaluators. *arXiv preprint arXiv:2206.05802*, 2022.

Jérémy Scheurer, Jon Ander Campos, Tomasz Korbak, Jun Shern Chan, Angelica Chen, Kyunghyun Cho, and Ethan Perez. Training language models with language feedback at scale. *arXiv preprint arXiv:2303.16755*, 2023.

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. doi: 10.52202/075280-0377.

Katherine Tian, Eric Mitchell, Allan Zhou, Archit Sharma, Rafael Rafailov, Huaxiu Yao, Chelsea Finn, and Christopher D. Manning. Just ask for calibration: Strategies for eliciting calibrated confidence scores from language models fine-tuned with human feedback. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 5433–5442, 2023. doi: 10.18653/v1/2023.emnlp-main.330.

Gladys Tyen, Hassan Mansoor, Victor Carbune, Peter Chen, and Tony Mak. Llms cannot find reasoning errors, but can correct them given the error location. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 13894–13908, 2024. doi: 10.18653/v1/2024.findings-acl.826.

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. *arXiv preprint arXiv:2211.14275*, 2022.

Yubo Wang, Xiang Yue, and Wenhu Chen. Critique fine-tuning: Learning to critique is more effective than learning to imitate. In *Second Conference on Language Modeling*, 2025.

Sean Welleck, Ximing Lu, Peter West, Faeze Brahman, Tianxiao Shen, Daniel Khashabi, and Yejin Choi. Generating sequences by learning to self-correct. In *The Eleventh International Conference on Learning Representations*, 2023.

Yixuan Weng, Minjun Zhu, Fei Xia, Bin Li, Shizhu He, Shengping Liu, Bin Sun, Kang Liu, and Jun Zhao. Large language models are better reasoners with self-verification. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 2550–2575, 2023. doi: 10.18653/v1/2023.findings-emnlp.167.

Zhihui Xie, Jie Chen, Liyu Chen, Weichao Mao, Jingjing Xu, and Lingpeng Kong. Teaching language models to critique via reinforcement learning. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 68559–68577. PMLR, 2025.

Miao Xiong, Zhiyuan Hu, Xinyang Lu, Yifei Li, Jie Fu, Junxian He, and Bryan Hooi. Can LLMs express their uncertainty? an empirical evaluation of confidence elicitation in LLMs. In *The Twelfth International Conference on Learning Representations*, 2024.

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.

Xiaoying Zhang, Yipeng Zhang, Hao Sun, Kaituo Feng, Chaochao Lu, Chao Yang, and Helen Meng. Critique-GRPO: Advancing LLM reasoning with natural language and numerical feedback. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026.

A Additional Method and Experimental Details

A.1 Full Critique-Component Results

Figure 3 summarizes the critique-component and diagnosis-only results visually. Table 2 provides the corresponding numerical values for every component and benchmark–solver–protocol setting.

Table 2: **Critique-component results.** Panel A reports correction and feedback tokens relative to the full critique within each cell (mean $\pm$ standard error over nine non-degenerate cells). Panel B reports diagnosis-only operating points; ΔAcc is the realized percentage-point change. †AIME–Llama is exploratory because its correct cohort contains only two histories.

(a) Correction and token use relative to full critique

Component	Correction / full (%)	Tokens / full (%)	Qwen correction / full (%)	Llama correction / full (%)
Verdict	42.56 ± 8.53	3.83 ± 0.36	28.62	70.43
Location	62.28 ± 6.94	14.55 ± 1.03	49.40	88.05
Diagnosis	98.81 ± 4.85	30.94 ± 1.84	99.28	97.88

(b) Diagnosis-only operating points

Benchmark–solver	Protocol	Initial accuracy (%)	$r_{w\to c}$ (%)	$r_{c\to w}$ (%)	ΔAcc (pp)
AIME–Qwen	Solver–critic	25.16	10.29	2.19	$+7.15$
AIME–Qwen	Self-refinement	25.16	2.21	2.03	$+1.14$
AIME–Qwen	Two-model debate	25.16	3.05	2.97	$+1.53$
AIME–Llama†	Solver–critic	2.60	0.33	28.12	-0.41
AIME–Llama†	Self-refinement	2.60	0.50	43.75	-0.65
AIME–Llama†	Two-model debate	2.60	0.00	21.88	-0.57
MATH–Qwen	Solver–critic	85.51	32.12	0.99	$+3.81$
MATH–Qwen	Self-refinement	85.51	9.64	0.44	$+1.02$
MATH–Qwen	Two-model debate	85.51	11.02	0.60	$+1.08$
MATH–Llama	Solver–critic	46.88	8.83	25.68	-7.35
MATH–Llama	Self-refinement	46.88	8.04	25.34	-7.61
MATH–Llama	Two-model debate	46.88	7.47	24.20	-7.38

Table 3: **Selected paired comparisons for DiSel-Consensus.** Differences are measured as Consensus minus the comparison selector. Confidence intervals and p-values are obtained from paired bootstrap tests clustered by problem.

Comparison	Outcome	Difference (pp)	95% CI	p
Consensus vs. Always revise	ΔAcc	$+0.880$	$[-0.004, 1.698]$	.0508
Consensus vs. Pairwise	ΔAcc	$+0.633$	$[0.019, 1.306]$	.0438
Consensus vs. Pointwise	ΔAcc	$+0.345$	$[-0.234, 0.998]$	.2730
Consensus vs. Pointwise	$c\to w$	-1.079	$[-2.143, -0.316]$	$<.0002$

A.2 Robustness controls

Verdict, location, diagnosis, and full critique expose increasingly detailed subsets of the same underlying record. Repeating the ablation under solver–critic, self-refinement, and two-model debate tests protocol dependence; an empty-critique condition measures resampling effects. GenLP, Pairwise, and Pointwise operate on the same frozen candidate pair but vary confidence elicitation and cost. Leave-one-group-out thresholding prevents calibration on the evaluated year or subject, and freezing the initial answer, diagnosis, and revision removes generation variance from selector comparisons.

B Prompt Design

We reproduce the prompt templates used in the experiments below. Braced terms denote runtime substitutions. For Qwen experiments, the AIME templates include the shown `/no_think` and

/think control lines; the corresponding lines are removed for AIME–Llama. For MATH–Qwen, /no_think is prepended to solver, self-refinement, debate, and revision system prompts, and /think is prepended to the critic system prompt. MATH–Llama uses the MATH-500 templates as printed.

Critique-component convention. Although the full critique was generated using a structured schema, our component ablation evaluates only three isolated components: verdict, error location, and diagnosis. Corrective hints and alternative approaches are not treated as separate experimental components. They appear only as part of the full-critique condition.

B.1 AIME 2020–2025 PROMPTS

Initial solver: system prompt.

```
/no_think
You are the initial solver in a frozen mathematical-reasoning
↪ experiment. Solve the
AIME problem independently. Show enough reasoning that a later critic
↪ can identify the
first consequential error. Do not ask for feedback or use external
↪ tools. AIME answers
are integers from 000 through 999. End with exactly one line in this
↪ format:
FINAL_ANSWER: ddd
```

Initial solver: user prompt.

```
Solve this AIME problem:
{question}
```

Solver–critic: critic system prompt.

```
/think
You are the critic in a controlled solver/critic experiment. Analyze
↪ the problem and
the frozen initial solver response. You are not given the reference
↪ answer. Do not
submit an independent AIME answer and do not use FINAL_ANSWER or a
↪ boxed final answer.
First decide one bit:
- OBJECT: you found a specific consequential error.
- CONFIRM: an independent derivation agrees and you found no specific
↪ error.
Then return exactly five XML fields. If the verdict is OBJECT, the
↪ other four fields
must form one accurate critique of the earliest consequential error.
↪ Local mathematical
values are allowed when needed to explain that error, but stop before
↪ deriving or
stating the final AIME answer. If the verdict is CONFIRM, put NONE in
↪ the other fields.
Component boundaries are strict:
- verdict: exactly OBJECT or CONFIRM.
- error_location: identify or quote the earliest consequential step;
↪ do not explain it.
- diagnosis: explain why that step is invalid; do not give a repair
↪ procedure.
- corrective_hint: give a concise repair principle; do not work the
↪ full solution.
```

```
- alternative_approach: outline a genuinely different checking or
↪ solution route, but
  stop before deriving a final numerical answer.
Do not add Markdown fences, extra XML fields, or text outside these
↪ tags:
<verdict>...</verdict>
<error_location>...</error_location>
<diagnosis>...</diagnosis>
<corrective_hint>...</corrective_hint>
<alternative_approach>...</alternative_approach>
```

Solver–critic: critic user prompt.

```
<problem>
{question}
</problem>
<initial_solver_response>
{initial_solution}
</initial_solver_response>
Return only the five tagged critique fields. Do not provide an
↪ independent answer.
```

Self-refinement: system prompt.

```
/no_think
You are performing the feedback step of self-refinement on your own
↪ frozen AIME
solution. Inspect that solution rigorously, but do not replace it
↪ with a new independent
solution and do not state a final AIME answer.
Return exactly five XML fields. If you find a consequential error,
↪ use OBJECT and make
the remaining fields a precise critique of the earliest such error.
↪ If you find no
specific error after checking, use CONFIRM and put NONE in the other
↪ fields.
Component boundaries:
- verdict: exactly OBJECT or CONFIRM.
- error_location: identify or quote the earliest consequential step
↪ without explaining it.
- diagnosis: explain why that step is invalid without giving a repair
↪ procedure.
- corrective_hint: give a concise repair principle without completing
↪ the solution.
- alternative_approach: outline a different checking route without
↪ deriving a final answer.
Return no Markdown fences or text outside these tags:
<verdict>...</verdict>
<error_location>...</error_location>
<diagnosis>...</diagnosis>
<corrective_hint>...</corrective_hint>
<alternative_approach>...</alternative_approach>
```

The self-refinement conversation then uses the following user message, the frozen initial response as an assistant message, and the final user instruction.

```
Solve the following AIME problem. End with exactly one `FINAL_ANSWER:
↪ ddd` line.
<problem>
{question}
```

```
</problem>
[assistant]
{initial_solution}
[user]
Critique your frozen response as the feedback phase of
↪ self-refinement. Return only the
five required XML fields and do not provide a final answer.
```

Two-model debate: Agent B system prompt.

```
/no_think
You are Agent B in a regular two-agent mathematical debate. Agent A
↪ has already proposed
a frozen AIME solution. Independently inspect Agent A's argument and
↪ respond with a
structured challenge or confirmation. Do not take over as a
↪ replacement solver and do
not state a final AIME answer.
Return exactly five XML fields. If you find a consequential error,
↪ use OBJECT and make
the remaining fields a precise critique of the earliest such error.
↪ If you find no
specific error after checking, use CONFIRM and put NONE in the other
↪ fields.
Component boundaries:
- verdict: exactly OBJECT or CONFIRM.
- error_location: identify or quote the earliest consequential step
↪ without explaining it.
- diagnosis: explain why that step is invalid without giving a repair
↪ procedure.
- corrective_hint: give a concise repair principle without completing
↪ the solution.
- alternative_approach: outline a different checking route without
↪ deriving a final answer.
Return no Markdown fences or text outside these tags:
<verdict>...</verdict>
<error_location>...</error_location>
<diagnosis>...</diagnosis>
<corrective_hint>...</corrective_hint>
<alternative_approach>...</alternative_approach>
```

Two-model debate: Agent B user prompt.

```
<problem>
{question}
</problem>
<agent_a_solution>
{initial_solution}
</agent_a_solution>
Respond to Agent A with only the five required XML critique fields.
↪ Do not provide a
final answer.
```

Revision solver: system prompt.

```
/no_think
You are the revision solver in a controlled mathematical-reasoning
↪ experiment. You
```

```
receive the problem, your frozen initial attempt, and one critic
↪ intervention. The
intervention may be accurate, absent, or wrong.
Re-derive any challenged step from first principles before deciding
↪ whether to revise.
If your derivation reproduces the original result, show that check
↪ and keep it. Do not
change an answer merely because it was challenged. If you change the
↪ solution, provide
the complete corrected derivation rather than simply accepting the
↪ critic's assertion.
For a neutral or confirming intervention, independently recheck the
↪ original derivation.
Do not mention the experiment. AIME answers are integers from 000
↪ through 999. End with
exactly one line in this format:
FINAL_ANSWER: ddd
```

Revision solver: user prompt.

```
<problem>
{question}
</problem>
<frozen_initial_solution>
{initial_solution}
</frozen_initial_solution>
<critique_intervention>
{challenge_text}
</critique_intervention>
Re-derive the challenged step from scratch, then defend or revise the
↪ solution and
return your final answer.
```

B.2 MATH-500 PROMPTS

Initial solver: system prompt.

```
You are the initial solver in a controlled mathematical-reasoning
↪ experiment. Solve the
problem carefully and show a coherent derivation. Do not mention the
↪ experiment.
End with exactly one final line in this format, placing the complete
↪ answer inside the
box even when it is symbolic, textual, a tuple, an interval, or a set:
FINAL_ANSWER: \boxed{answer}
```

Initial solver: user prompt.

```
Solve this problem.
<problem>
{question}
</problem>
```

Solver–critic: critic system prompt.

```
You are the critic in a solver-versus-critic experiment. Inspect the
↪ frozen solver
```

```
response rigorously. Your only job is to challenge or confirm that
↪ response; do not take
over as an independent solver and do not state a final answer.
Return exactly five XML fields. If you find a consequential error,
↪ use OBJECT and make
the remaining fields a precise critique of the earliest such error.
↪ If you find no
specific error after checking, use CONFIRM and put NONE in the other
↪ fields.
Component boundaries:
- verdict: exactly OBJECT or CONFIRM.
- error_location: identify or quote the earliest consequential step
↪ without explaining it.
- diagnosis: explain why that step is invalid without giving a repair
↪ procedure.
- corrective_hint: give a concise repair principle without completing
↪ the solution.
- alternative_approach: outline a different checking route without
↪ deriving a final answer.
Return no Markdown fences or text outside these tags:
<verdict>...</verdict>
<error_location>...</error_location>
<diagnosis>...</diagnosis>
<corrective_hint>...</corrective_hint>
<alternative_approach>...</alternative_approach>
```

Solver–critic: critic user prompt.

```
<problem>
{question}
</problem>
<frozen_solver_response>
{initial_solution}
</frozen_solver_response>
Return only the five required XML critique fields. Do not provide a
↪ final answer.
```

Self-refinement: system prompt.

```
You are performing the feedback step of self-refinement on your own
↪ frozen mathematical
solution. Inspect that solution rigorously, but do not replace it
↪ with a new independent
solution and do not state a final answer.
Return exactly five XML fields. If you find a consequential error,
↪ use OBJECT and make
the remaining fields a precise critique of the earliest such error.
↪ If you find no
specific error after checking, use CONFIRM and put NONE in the other
↪ fields.
Component boundaries:
- verdict: exactly OBJECT or CONFIRM.
- error_location: identify or quote the earliest consequential step
↪ without explaining it.
- diagnosis: explain why that step is invalid without giving a repair
↪ procedure.
- corrective_hint: give a concise repair principle without completing
↪ the solution.
- alternative_approach: outline a different checking route without
↪ deriving a final answer.
Return no Markdown fences or text outside these tags:
```

```
<verdict>...</verdict>
<error_location>...</error_location>
<diagnosis>...</diagnosis>
<corrective_hint>...</corrective_hint>
<alternative_approach>...</alternative_approach>
```

The self-refinement conversation uses the MATH-500 initial user prompt, the frozen initial response as an assistant message, and then:

```
Critique your frozen response as the feedback phase of
↪ self-refinement. Return only the
five required XML fields and do not provide a final answer.
```

Two-model debate: Agent B system prompt.

```
You are Agent B in a regular two-agent mathematical debate. Agent A
↪ has already proposed
a frozen solution. Inspect Agent A's argument and respond with a
↪ structured challenge or
confirmation. Do not take over as a replacement solver and do not
↪ state a final answer.
Return exactly five XML fields. If you find a consequential error,
↪ use OBJECT and make
the remaining fields a precise critique of the earliest such error.
↪ If you find no
specific error after checking, use CONFIRM and put NONE in the other
↪ fields.
Component boundaries:
- verdict: exactly OBJECT or CONFIRM.
- error_location: identify or quote the earliest consequential step
↪ without explaining it.
- diagnosis: explain why that step is invalid without giving a repair
↪ procedure.
- corrective_hint: give a concise repair principle without completing
↪ the solution.
- alternative_approach: outline a different checking route without
↪ deriving a final answer.
Return no Markdown fences or text outside these tags:
<verdict>...</verdict>
<error_location>...</error_location>
<diagnosis>...</diagnosis>
<corrective_hint>...</corrective_hint>
<alternative_approach>...</alternative_approach>
```

Two-model debate: Agent B user prompt.

```
<problem>
{question}
</problem>
<agent_a_solution>
{initial_solution}
</agent_a_solution>
Respond to Agent A with only the five required XML critique fields.
↪ Do not provide a
final answer.
```

Revision solver: system prompt.

```
You are the revision solver in a controlled mathematical-reasoning
↪ experiment. You
receive the problem, your frozen initial attempt, and one feedback
↪ intervention. The
intervention may challenge the solution, confirm it, or be absent.
Re-derive any challenged step from first principles before deciding
↪ whether to revise.
If your derivation reproduces the original result, show that check
↪ and keep it. Do not
change an answer merely because it was challenged. If you change the
↪ solution, provide
the complete corrected derivation rather than simply accepting the
↪ feedback assertion.
For absent or confirming feedback, independently recheck the original
↪ derivation.
Do not mention the experiment. End with exactly one final line in
↪ this format, placing
the complete answer inside the box even when it is symbolic, textual,
↪ a tuple, an
interval, or a set:
FINAL_ANSWER: \boxed{answer}
```

Revision solver: user prompt.

```
<problem>
{question}
</problem>
<frozen_initial_attempt>
{initial_solution}
</frozen_initial_attempt>
<feedback>
{feedback}
</feedback>
Produce the complete checked solution and the required final-answer
↪ line.
```