CONTEXTUAL DRAG: HOW ERRORS IN THE CONTEXT AFFECT LLM REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Many self-improvement pipelines for large language models (LLMs) reuse prior attempts as context, relying on verification or critique to extract useful reasoning from flawed attempts. We study *contextual drag*: failed attempts in context bias subsequent generations toward structurally similar errors, often persisting even when models recognize them as wrong. We focus on *self-drag*, where a model conditions on its own incorrect drafts, mirroring iterative self-refinement. Across 10 open-weight reasoning models and 8 reasoning tasks, a single incorrect draft drops average pass@1 accuracy by around 25 points on problems where direct reasoning produces both successes and failures. For GPT-OSS-20B, which has severe single-step self-drag, iterative refinement compounds this effect into *self-deterioration*. Explicit error signals, from either external labels or the model's own correct verification, do not fully eliminate the effect, suggesting that correct recognition is not sufficient to reset the draft's influence. A Game of 24 case study using tree edit distance shows that subsequent generations are structurally biased toward the in-context draft, not just less accurate.

Finally, filtering the draft before reuse reduces contextual drag on every model we test, by a model-dependent amount and without training or external labels; applied recursively, it turns GPT-OSS-20B's self-deterioration on competition math into steady improvement. These results suggest that robust self-improvement requires more than reflecting on mistakes; it requires mechanisms for context reset, independent attempts, and selective reuse of prior reasoning.

1 INTRODUCTION

Modern large language model (LLM) systems attempt to improve outputs by reusing previous attempts as context. In iterative self-refinement, a model critiques and revises its own prior response (Madaan et al., 2023); in multi-agent aggregation workflows, later agents inspect and combine earlier solutions (Wang et al., 2024). These pipelines share a common design: previous attempts are not discarded, but are reused as context for subsequent improvement.

This reuse of context is useful but also risky. Prior work has shown that LLMs are highly sensitive to information in the prompt: misleading hints, user errors, or framing can override the model's own knowledge and induce sycophantic behavior (Turpin et al., 2023; Sharma et al., 2024; Chen et al., 2025; Zhu et al., 2025b; Simhi et al., 2026). These findings suggest that models do not always treat context as neutral evidence; salient in-context information can exert a strong influence on generation.

In self-improvement pipelines, this tendency exposes a more structured failure mode. Although extended reasoning and explicit verification might be expected to help models reject bad context (Zhou et al., 2024; Huang et al., 2025), it remains unclear whether such "local judgment" fully resets the downstream influence of contextual errors. That is, a draft that is accurately rejected could still shape the model's later reasoning structure, *dragging* it toward similar errors.

Therefore, the following research questions are of interest:

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.

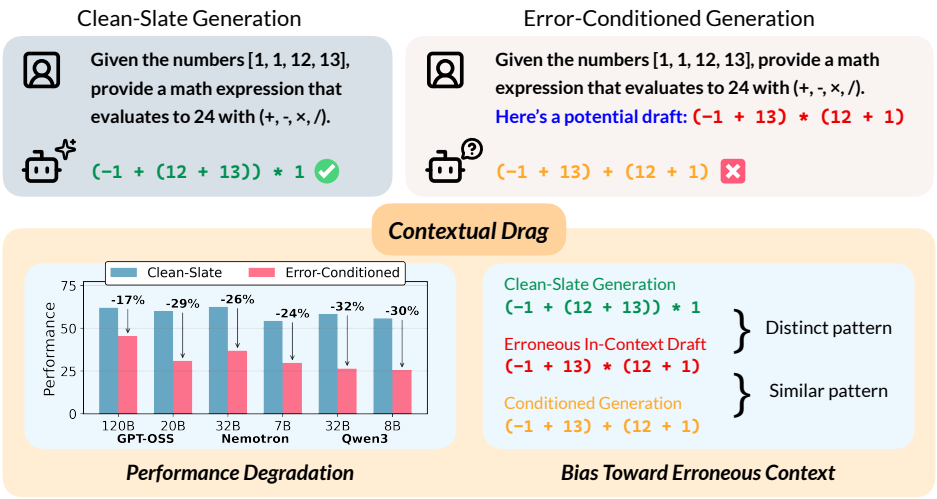

Figure 1: **Contextual drag** is characterized by the performance drop from clean-slate generation, where the model generates with no additional context, to error-conditioned generation, where the model generates conditioned on incorrect drafts in the context. Contextual drag not only manifests as performance degradation but also as a bias in reasoning patterns toward erroneous context.

(RQ1) How strongly do incorrect drafts in context degrade subsequent reasoning in the self-refinement setting with explicit verification? How is the effect manifested in the observable structure of generated solutions, beyond final-answer accuracy?

(RQ2) Is the degradation merely a failure to detect errors, or does it persist under *error signals* (explicit indications that the draft is wrong) from external labels or self-verification?

(RQ3) Can context-control interventions, such as filtering, rewriting, or reset reduce the influence and restore self-refinement?

Main Findings

We identify *contextual drag* in large reasoning models, whereby conditioning on failed reasoning attempts in context biases subsequent generations toward structurally similar errors. Prior work locates the failure of self-correction in error detection (Huang et al., 2024; Tyen et al., 2024); we show that degradation persists after detection succeeds. We define contextual drag generally: failed drafts may come either from the model itself or from other models. Our main study focuses on *self-drag*, where a model is conditioned on its own prior incorrect drafts, because this setting is directly relevant to iterative self-refinement. We report complementary *cross-drag* results in Section B, where in-context drafts can come from other models.

Contextual drag is widespread under self-drag (Section 2.1). We evaluate on the test-time-scaling-relevant subset: problems for which clean-slate sampling produces both correct and incorrect attempts. On this subset, a single incorrect self-generated draft reduces average pass@1 by roughly 25 percentage points across 10 open-weight reasoning models and 8 tasks covering math, science, code, and puzzle solving. The effect is qualitatively strongest for smaller models. Larger models and higher thinking budgets attenuate the effect, but do not eliminate it. Cross-drag results in Section B show that the effect is not limited to open-weight models: frontier proprietary models also showsmall drops when conditioned on incorrect drafts from other models.

Contextual drag can compound under naive iterative refinement pipelines (Section 2.2). When GPT-OSS-20B, a model with severe single-step self-drag, iteratively refines its previous attempt, it shows *self-deterioration*; Qwen3-32B, with nearly identical single-step drag, stays roughly flat (-3.5 points by round 8; Section F.5). Thus, contextual drag poses a significant failure mode for reliable self-improvement: when flawed intermediate reasoning is repeatedly reused as raw context, more inference-time compute can amplify errors rather than correcting them.

A Game of 24 case study using tree edit distance (Section 2.3) shows that, under contextual drag, new solutions are structurally closer to the incorrect draft than clean-slate solutions are.

Error signals do not reliably reset context (Sections 3.1 and 3.2). Explicitly labeling the draft as incorrect does not restore clean-slate performance in most cases. Even when the error signal is produced by the model's own verification, the recovery is strongly model-dependent. Some families, such as Nemotron OpenMathReasoning and Gemma 4, partially recover, while GPT-OSS-20B and the Qwen3 family remain strongly degraded. This separates two capabilities that are often conflated in self-improvement pipelines: detecting erroneous context and resetting or selectively reusing contents to prevent errors from contaminating the continuation.

Finally, **filtering the context before reuse reduces contextual drag** (Section 4.1). Filtering the draft before the solver sees it improves over naive reuse on every model we test, by a model-dependent amount; rewriting it helps on most models but can regress on smaller ones. In particular, recursively applying simple context filtering turns GPT-OSS-20B's self-deterioration into steady improvement, and yields a modest gain for Qwen3-32B over a roughly flat baseline ($+2.1$ points over round 0). These results suggest that prior attempts can be useful, but only when their influence is controlled through context reset, isolation of independent attempts, and selective reuse of validated reasoning.

2 Empirical Evaluation of Contextual Drag in Reasoning Models

In this section, we evaluate contextual drag in the *self-drag* setting, where models condition on their own erroneous drafts. This setting isolates the failure mode relevant to iterative self-refinement: a model attempts to improve by reusing its own previous flawed reasoning. We first show that self-drag causes substantial pass@1 degradation across models and tasks (Section 2.1), then show that this degradation can compound under iterative refinement (Section 2.2), and finally quantify the structural component of the effect using tree edit distance on Game of 24 (Section 2.3). Complementary cross-drag results, where drafts come from other models, are reported in Section B.

2.1 Setup and Main Results

We compare three conditions (illustrated in Figure 7, Section A). Under DIRECT, the model solves the problem with no additional context. Under 1F/2F, it is additionally given one or two of its own incorrect drafts and asked to verify each draft and output a verdict before writing a new solution (verify-then-solve). The drop from DIRECT to 1F/2F measures contextual drag.

Models and benchmarks Our main evaluation uses 10 open-weight reasoning models from five families: GPT-OSS-20B/120B (OpenAI, 2025a), OpenReasoning-Nemotron-7B/32B (Ahmad et al., 2025b;a; Moshkov et al., 2025), Qwen3-8B/32B (Yang et al., 2025), Gemma4-E4B/31B (Team, 2026) and DeepSeek R1-distilled Llama-8B/Qwen-7B (DeepSeek-AI, 2025). We evaluate on problems from four competition math datasets: AIME24/25 and HMMT24/25 (Balunović et al., 2025), two general QA benchmarks: GPQA (Rein et al., 2024) and MMLU-Redux 2.0 (Gema et al., 2024), one code reasoning benchmark: CRUXEval-I (Gu et al., 2024), and one puzzle dataset (Game of 24).

Evaluation setup We focus on *self-drag*, where the evaluated model is conditioned on its own attempts. For each (model, task) pair, we first run 16 clean-slate generations under DIRECT (no additional context). We evaluate contextual drag on the subset of problems for which these clean-slate samples contain at least two correct and at least two incorrect attempts. This criterion removes saturated and unsolved instances. On the retained problems the model can solve the problem but does not do so reliably, which is where self-improvement and test-time scaling matter most. Sizes of the filtered subsets are reported in Table 2. Evaluation setup details are deferred to Section A.

Results Figure 2 shows that self-drag causes widespread and substantial degradation across the model suite. Averaged over the 8 tasks, conditioning on a single self-generated incorrect draft reduces pass@1 by roughly 25 percentage points. The effect is larger for GPT-OSS-20B, Qwen3, and the smaller R1-distilled models. Stronger models such as GPT-OSS-120B, Nemotron-32B, and Gemma4-31B also suffer from noticeable degradation. Adding a second incorrect draft generally amplifies the drop, suggesting that verification does not simply average away or cancel the influence

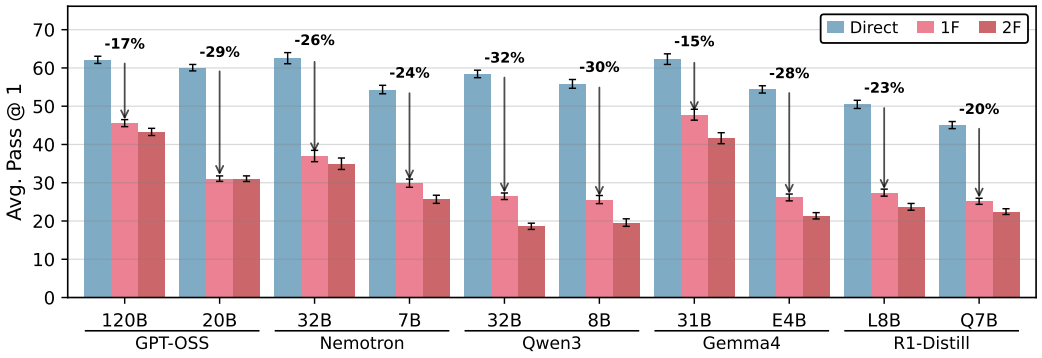

Figure 2: **Self-drag pass@1 across 10 open-weight reasoning models and 8 tasks.** DIRECT is clean-slate; 1F/2F condition on one or two of the model's own incorrect drafts. Conditioning on incorrect drafts induces consistent, substantial drops. Two drafts generally lead to larger degradation. Full results are deferred to Table 3 in appendix. Error bars reflect one standard deviation.

of multiple erroneous drafts. These results show that, on problems where clean-slate sampling reveals both successes and failures, incorrect attempts in context can suppress the model's reasoning capability. Majority voting over conditioned samples does not close this gap (Section A.6). We provide additional results on code generation (LiveCodeBench v6) in Section A.5.

Ablations In Section C, we test alternative explanations. A larger thinking budget attenuates the drop but does not remove it (Section C.1); reordering the prompt, using a matched prompt, or removing the verification requirement changes its magnitude but does not reliably restore DIRECT (Sections C.2 and C.3); and correct drafts help substantially, while incorrect drafts can erase those gains (Section C.4). Contextual drag is therefore not simply a byproduct of reasoning effort or a general inability to use context.

2.2 SELF-DETERIORATION UNDER NAIVE ITERATIVE REFINEMENT

In iterative self-refinement, the model repeatedly conditions on its own previous output. Prior successes of self-improvement pipelines often rely on some form of guidance, selection, or feedback, whereas unguided self-correction often fails to improve reasoning and can even degrade performance (Huang et al., 2024; Xu et al., 2024; Lee et al., 2025). Our setting stresses this unguided case: the model receives its own previous attempt as context, verifies it, and then produces a new solution. Conditioning on incorrect drafts in Section 2.1 isolates the failure mode. We then study the realistic deployment case, where previous attempts are iteratively reused and refined regardless of their correctness, and show that contextual drag compounds for GPT-OSS-20B. Our results suggest that flawed reasoning in context may contaminate later reasoning and become amplified throughout the iterative process. We use the verify-then-solve setup standard in self-refinement pipelines.

We evaluate this dynamic on GPT-OSS-20B, a model with severe single-step self-drag. Starting from a clean-slate response at round 0, each subsequent round conditions on the previous round's response as the draft and asks the model to verify and re-solve. As shown in Figure 3, standard iterative refinement is brittle. Accuracy declines across refinement rounds on all four math benchmarks, yielding *self-deterioration* rather than self-improvement. Meanwhile, the trend is not universal. Across four more models, iterative outcomes range from flat to improving and track single-step drag severity (Section F.5). This compounding behavior is reminiscent of hallucination snowballing (Zhang et al., 2024). This result does not imply that prior attempts are never useful. Instead, it highlights the implication of contextual drag for unguided self-refinement pipelines that reuse prior attempts without additional context control: additional inference naively conditioned on prior reasoning can amplify earlier errors rather than correct them. The refinement curve after mitigation in Figure 3 previews the complementary lesson developed in Section 4.1: refinement becomes more reliable when prior attempts are pre-processed, so that useful information can be selectively reused with erroneous reasoning properly discarded or isolated.

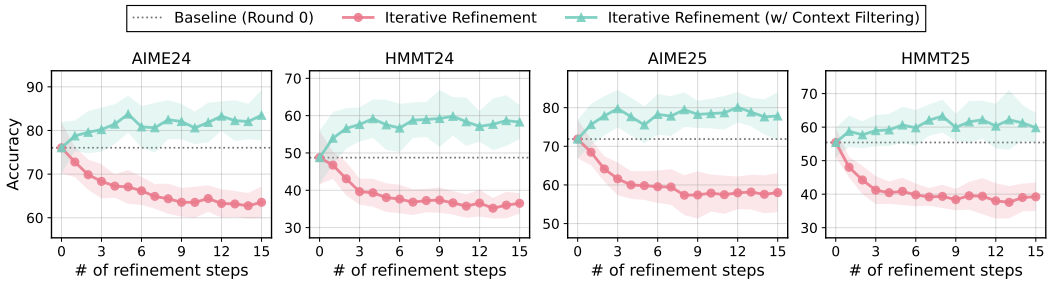

Figure 3: **Self-deterioration vs. recursive context filtering** (GPT-OSS-20B, 4 math tasks, mean across 16 trajectories with shaded area reflecting 1 std.). Standard iterative refinement (red) declines across rounds while recursive filtering (green, see Section 4.1) flips the trajectory into improvement. The dotted line is the DIRECT (round-0) policy. Recursive filtering is fully model-internal: it uses no oracle or ground-truth labels.

2.3 STRUCTURAL CHARACTERIZATION OF CONTEXTUAL DRAG ON GAME OF 24

The pass@1 drops in Section 2.1 and the self-deterioration dynamics in Section 2.2 show that erroneous in-context drafts harm performance. We next ask how the draft shapes the model's response beyond correctness. Compared with clean-slate generations, do conditioned generations more closely resemble the failed draft in answer structure?

We study this question on Game of 24. Given four integers, the problem asks for an arithmetic expression using each number exactly once and operations from $\{+, -, \times, \div\}$ that evaluates to 24. Since each solution can be parsed into an expression tree, with input numbers as leaves and operations as internal nodes, we measure structural similarity between a response and the in-context draft using tree edit distance (TED) (Zhang & Shasha, 1989). For each problem, we compare the incorrect draft against two response sets: clean-slate generations under DIRECT and conditioned generations under 1F/2F. As shown in Figure 4 (left), conditioned 1F generations are consistently closer in TED to the in-context incorrect draft than clean-slate generations are. The same trend holds under 2F (Figure 4, right), where we evaluate the minimum among the TED between the solution and the two drafts. This suggests that contextual drag is not only an answer-level accuracy drop but also more similar solution structure. More details on the analysis and ablations on prompt designs are deferred to Section D.

Conditioned generations are thus not only more likely to be wrong but also structurally closer to the draft. This helps explain why error labels and verification may not be enough for recovery (Section 3): they can lead the model to judge the draft as wrong, but the draft stays in context and can still shape the new solution. This motivates the context-side interventions in Section 4.1.

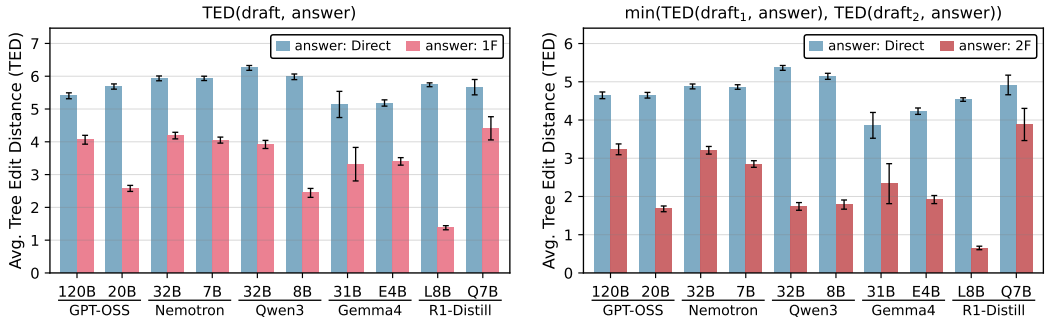

Figure 4: **Conditioned generations are structurally closer to the in-context draft (Game of 24).** Mean tree edit distance (TED) between each generation and the incorrect draft, comparing clean-slate (DIRECT) generations to conditioned (1F, left; 2F, right) generations. Lower TED indicates stronger structural similarity to the draft. Error bars reflect one standard deviation.

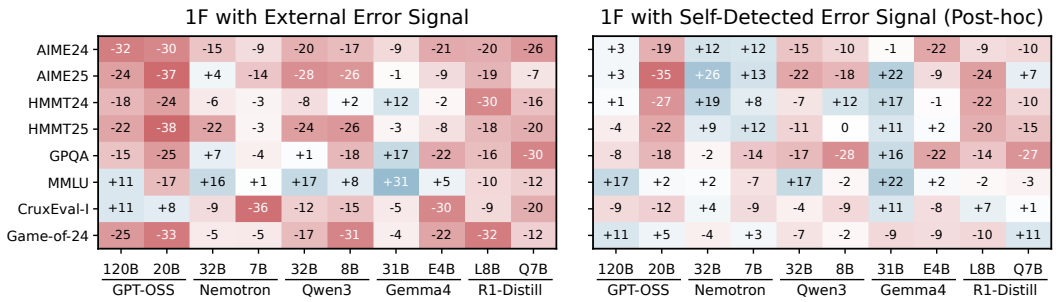

Figure 5: **Error signals do not eliminate contextual drag.** The 1F performance change relative to DIRECT when (left) the prompt explicitly states the draft is incorrect (**external**, Section 3.1) and (right) only on trajectories where the model itself correctly flags the draft (**self-detected**, Section 3.2). Self-detected error helps Nemotron and Gemma 4-31B, but GPT-OSS-20B and the Qwen3 family remain strongly degraded.

3 CONTEXTUAL DRAG UNDER ERROR SIGNALS

A natural hypothesis is that contextual drag is primarily a verification failure: once the draft is labeled or recognized as incorrect, the model should discount it when producing a new solution. Prior work has shown that intrinsic self-correction in LLMs is unreliable (Huang et al., 2024; Tyen et al., 2024; Stechly et al., 2023). We go further and show that the drag persists even when the error is correctly recognized, under both external signals (Section 3.1) and self-verification (Section 3.2).

3.1 EXTERNAL ERROR SIGNAL

We first test a prompt-level external error signal, where we reuse the 1F self-drag setup from Section 2.1 with a modified prompt that explicitly states that the draft has an incorrect final answer and instructs the model not to copy it. This removes the need for the model to infer incorrectness, isolating whether it can discount a draft once its incorrectness is known. As shown in Figure 5 (left), external labeling does not eliminate contextual drag. Pass@1 in 1F remains substantially below DIRECT on most model-task pairs[1]. We repeat the same TED analysis for the Game of 24 TED generations (see Figure 15) and see that conditioned generations remain structurally closer to the draft than clean-slate generations do. Thus, a prompt-level error signal can only change the model's stated treatment of the draft without reliably removing the draft's influence on the generated solution.

3.2 SELF-DETECTED ERROR SIGNAL (POST-HOC)

We next ask whether the drag persists when the model itself identifies the draft as wrong. We filter the 1F generations in Section 2.1 to retain only trajectories where the model emits an accurate "Incorrect" verdict for the draft[2]. This is a post-hoc analysis which effectively assumes access to an oracle filter, hence biasing toward stronger performance. As shown in Figure 5 (right), correct self-verification produces strongly model-dependent outcomes. For Nemotron and Gemma 4-31B, the subset of trajectories with correctly self-detected errors substantially recovers performance and often reaches or exceeds DIRECT on many tasks. In contrast, GPT-OSS-20B and the Qwen3 family remain substantially degraded despite correctly emitting an "Incorrect" verdict. The TED comparison on Game of 24 mirrors this split (Figure 15), with structural similarity to the draft persisting for the models that do not recover. Because this analysis conditions on the model's own verdict, it shows that correct verdicts co-occur with persistent degradation; Section 3.1 provides the cleaner test, since the signal there does not depend on the model's own verdict.

These results separate two capabilities that are often conflated in self-improvement pipelines. The first is *error detection*, identifying that a prior attempt in the context is wrong, and the second is *context reset*, preventing the rejected content from shaping future reasoning. Correct recognition is useful,

[1]MMLU is an outlier likely because flagging one multiple-choice option as false shrinks the search space.
[2]We verify verdicts via parsable structured tags in the model output. See Section G.2 for the format.

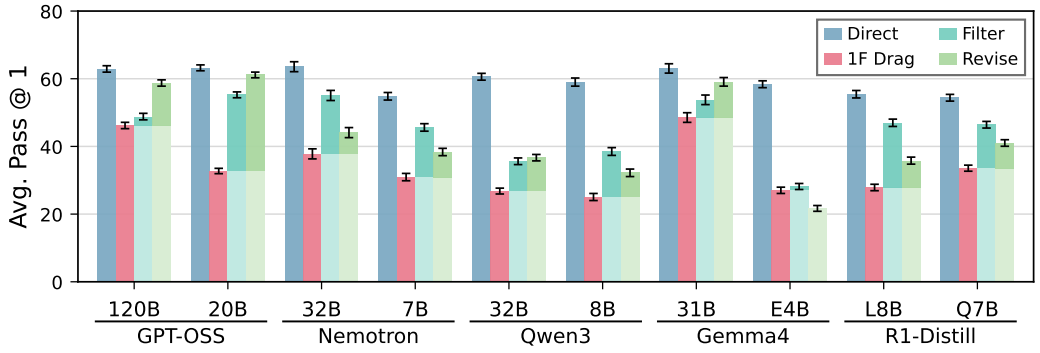

Figure 6: **Test-time context denoising reduces self-drag, by a model-dependent amount.** Per-model pass@1 (%) under DIRECT, 1F, and the two denoising pipelines FILTER and REVISE applied to the 1F draft, averaged over the eight tasks (Table 23). FILTER improves over 1F on every model; REVISE matches it on larger reasoners but regresses on Gemma4-E4B (Table 24). Per-model recovery of the gap ranges from 3.6% to 73.8% for FILTER and from -17.1% to 93.1% for REVISE. Error bars reflect $\pm$std.

but it is not sufficient to reset the draft's influence. This motivates interventions that operate directly on the context, rather than relying solely on an internal verdict to control downstream reasoning.

4 MITIGATING CONTEXTUAL DRAG THROUGH CONTEXT CONTROL

In Section 3, we show that erroneous drafts can degrade subsequent reasoning even when explicitly flagged or recognized as incorrect. This suggests that mitigation should target the reusable context itself, not only the model's verdict on it. Rather than discarding prior attempts entirely, the goal is to control their influence by extracting useful components while suppressing erroneous ones. We therefore study the effect of *context denoising* that modifies or isolates the drafts before passing them to the solver. We compare two test-time context denoising methods that can be seen as in-setting, single-turn adaptations of prior self-improvement work (Shinn et al., 2023; Madaan et al., 2023): FILTER, which extracts only the components of the draft that are consistent with a freshly generated strategy, and REVISE, which rewrites the draft into an improved solution before using it for solving. FILTER and REVISE are deliberately simple interventions rather than proposed state-of-the-art methods: they test whether controlling only the context, without changing the model, reduces the drag.

Together with the conditions of Sections 2 and 3, these methods cover six common strategies for handling prior attempts: reusing them raw, labeling them externally, verifying first, repairing before reuse (REVISE), selective reuse (FILTER), and discarding and resampling (DIRECT); Section F.7 maps each to prior work. Only selective reuse gives consistent single-step gains and flips iterative deterioration.

4.1 TEST-TIME CONTEXT DENOISING REDUCES DRAG

4.1.1 METHODS

Similar to the refine operation of Madaan et al. (2023), REVISE asks the model to rewrite the draft into an improved solution, then solves from the revised draft. This tests whether repairing the flawed context before reuse reduces contextual drag. However, revision does not fully isolate the erroneous draft: since the draft remains in context, the revised output may still inherit its influence.

FILTER, on the other hand, separates strategy generation from draft inspection. It first asks the model to generate a high-level strategy from the problem alone, without seeing the draft. It then uses this clean-slate strategy as a criterion for extracting only the correct and useful components of the draft. The solver conditions on this filtered context rather than the raw draft. Thus, FILTER implements both isolation and selective reuse by forming an independent plan before deciding which parts of the

prior attempt are safe to be carried forward. FILTER uses three calls per draft (strategy, filter, solve) and REVISE two (Section F.1).

4.1.2 RESULTS

Single-step denoising reduces drag on every model. Across the ten models in Figure 6, both denoising methods substantially reduce the average gap between 1F and DIRECT, reaching or exceeding the clean-slate performance on some model–task pairs (Table 23). Contextual drag can thus be reduced by controlling the context alone, and the recovery extent depends on the model. Simply discarding the draft is not a practical alternative: it gives up the gains from correct drafts (Section C.4) and requires knowing which drafts are wrong. We note that the recovery is not uniform. REVISE yields mixed results for some smaller models, occasionally regressing below 1F (e.g., Gemma4-E4B in Table 24), likely because the revision step remains directly conditioned on the erroneous draft. The same recovery pattern holds, and is in fact stronger, in the cross-drag setting (Figure 11).

FILTER yields more consistent improvement and is more directly aligned with selective reuse. The freshly generated strategy provides an external criterion for retaining useful draft components while discarding unreliable reasoning. FILTER combines draft-blind planning with draft filtering; we do not separate the contributions of the two components.

Recursive filtering turns deterioration into improvement. The strongest test of context denoising is in iterative refinement, where contextual drag previously compounded into self-deterioration. We repeat the self-refinement experiment from Section 2.2 with the FILTER approach applied before each solving inference so that only the filtered context gets passed to the solver.

As shown in Figure 3, GPT-OSS-20B that deteriorates under naive iterative self-refinement improves when the previous attempt is filtered before reuse. By round 8, recursive filtering yields a $+18$ to $+24$ percent point absolute gain over the unfiltered baseline across all four math benchmarks (Table 25). This setting, which reuses each previous attempt regardless of its correctness, is closest to how self-improvement pipelines are deployed. There, recursive filtering not only avoids deterioration but is also 6.4–10.2 points above the round-0 baseline by round 8 on all four competition math benchmarks. This result clarifies why prior self-improvement pipelines can succeed despite contextual drag. Past attempts can be useful, but only when the system controls their influence through context isolation, filtering, and selective reuse. Recursive filtering adds only moderate per-round inference overhead; we report the full timing breakdown in Section A.7 and additional results on Qwen3-32B in Table 27.

4.2 TRAINING-TIME FALLBACK IS NOT AN EFFECTIVE MITIGATION

As a complementary direction, we also evaluate *conditional fallback training* in Section G. This trains the model to verify the in-context draft and fall back to a clean-slate solution trajectory when the draft is judged incorrect. Conditional fallback training is not an effective mitigation: 1F gains come at a larger average 1T cost, and 1F remains below DIRECT. We report it as evidence that contextual drag is not easily trained away, and that inference-time context control is the more effective mitigation.

5 RELATED WORK

Recursive self-improvement Many LLM pipelines seek to improve outputs by reusing prior generations as context. These include iterative self-refinement, in which a model critiques and revises its own previous attempt (Madaan et al., 2023), verbal-reflection agents that maintain a textual memory of past failures to guide subsequent trials (Shinn et al., 2023), training-time bootstrapping that fine-tunes a model on its own correct rationales (Zelikman et al., 2022), and multi-agent aggregation in which later agents condition on earlier solutions (Wang et al., 2024; Venkatraman et al., 2025; Madaan et al., 2025; Li et al., 2025). These methods rely on prior attempts containing useful partial reasoning. Our work identifies a complementary failure mode: when prior attempts are wrong, indiscriminate reuse can bias later generations toward the failed attempt even after critique or verification.

Limits of self-correction Prior work has shown that unguided self-correction is often unreliable for reasoning tasks, and that refinement benefits from external feedback, checklists, or stronger guidance

(Huang et al., 2024; Stechly et al., 2023; Xu et al., 2024; Lee et al., 2025). A complementary line of evidence shows that LLMs struggle to localize their own reasoning errors, yet can correct those errors once an external signal indicates where they occur (Tyen et al., 2024). Contextual drag provides a more specific account of one way refinement can fail: a model may correctly identify a previous attempt as flawed, yet still fail to prevent the flawed trace from shaping future reasoning.

Agentic self-improvement Failure propagation also appears in full agent systems. Zhu et al. (2025a) study where LLM agents fail and how errors propagate across steps; Epperson et al. (2025) rely on human-mediated context reset to debug and steer multi-agent systems; and Zhao et al. (2024) curates past experience before reusing it. These works observe failure propagation or curate reused experience inside full agent harnesses; we isolate the underlying primitive: models do not reliably reset internally, even after correctly recognizing an error.

Misleading context, noisy rationales, and anchoring A broad line of work shows that LLMs are highly sensitive to misleading or salient prompt context. Noisy rationales and contextual distractors can degrade reasoning even under critique instructions (Zhou et al., 2024; Lee et al., 2026); biased hints can steer outputs through anchoring (Nguyen, 2024; Huang et al., 2025; O'Leary, 2025); and user errors can similarly induce sycophantic agreement (Sharma et al., 2024; Turpin et al., 2023; Chen et al., 2025; Zhu et al., 2025b). Min et al. (2022) find that models largely ignore label correctness in demonstrations of other problems; we study erroneous reasoning on the same problem, where the influence is strong, shapes solution structure, and persists after the error is recognized. Contextual drag concerns how rejected reasoning continues to shape later reasoning in systems that deliberately reuse prior attempts.

6 Discussion and Future Directions

Contextual drag exposes a context-control problem in self-improvement pipelines. Prior attempts can contain useful partial progress, but failed attempts can also remain influential after being rejected. Correct drafts help substantially (Section C.4), so reliable refinement should not discard context entirely, but should control how prior reasoning is reused. This view reconciles our findings with prior self-improvement work rather than contradicting it: successful pipelines include corrective structure such as feedback, selection, guidance, or role specialization, and self-critique helps when the reused context is controlled but fails when erroneous attempts are reused naively. While our Game of 24 analysis provides an observable structural measure of drag, richer measures are needed for open-ended reasoning domains.

We do not aim to provide a full mechanistic account of contextual drag. The phenomenon unfolds over long reasoning traces and manifests in downstream solution structure and refinement dynamics, so we focus on behavioral characterization and mitigation. Understanding the mechanisms by which models fail to reset from rejected context remains an important direction for future work.

7 Conclusion

We identify *contextual drag* as a failure mode of reasoning with flawed prior attempts in context. In the self-drag setting, where a model conditions on its own incorrect drafts, failed reasoning traces substantially degrade performance and bias later generations toward similar solution structures. The effect persists under external error labels and correct self-verification, showing that correct recognition is not sufficient to reset the draft's influence.

This directly challenges naive iterative refinement: reusing previous attempts indiscriminately can compound errors into self-deterioration. Yet prior attempts are not inherently harmful; filtering drafts before reuse can reverse this trend and restore improvement. Reliable self-improvement therefore requires context-control mechanisms that separate generation, verification, filtering, and final solving, rather than relying on a single model call to both critique and internally discount flawed context.

AI USE STATEMENT

We used LLMs for literature search, developing codebase (as coding agents), and polish writing for this paper. We also use a set of open source LLMs to generate the incorrect / correct draft solutions, which are used to curate the contextual drag evaluation sets.

ETHICS STATEMENT

This paper studies *contextual drag*: a failure mode where conditioning on an incorrect prior attempt can bias subsequent reasoning toward the same errors, even when incorrectness is signaled or recognized. By characterizing the error mode and by exploring mitigations (e.g., training and test-time strategies to reduce reliance on erroneous context), our work may improve the reliability of iterative self-improvement pipelines and downstream systems that depend on multi-step reasoning, verification, or tool use, potentially reducing silent error propagation in applications.

The primary potential risk would be dual-use: our analysis may help adversaries more deliberately steer models using misleading in-context "solutions" or craft stronger prompt-injection style attacks by exploiting anchoring dynamics. To mitigate this, we emphasize robustness-oriented evaluation, document limitations and failure cases, and frame proposed techniques as defenses that can be incorporated into safer prompting and training practices. Our experiments use standard benchmarks and model-generated traces, without collecting personal data.

REFERENCES

Wasi Uddin Ahmad, Somshubra Majumdar, Aleksander Ficek, Sean Narenthiran, Mehrzad Samadi, Jocelyn Huang, Siddhartha Jain, Vahid Noroozi, and Boris Ginsburg. Opencodereasoning-ii: A simple test time scaling approach via self-critique. *arXiv preprint arXiv:2507.09075*, 2025a.

Wasi Uddin Ahmad, Sean Narenthiran, Somshubra Majumdar, Aleksander Ficek, Siddhartha Jain, Jocelyn Huang, Vahid Noroozi, and Boris Ginsburg. Opencodereasoning: Advancing data distillation for competitive coding. *arXiv preprint arXiv:2504.01943*, 2025b.

Alon Albalak, Duy Phung, Nathan Lile, Rafael Rafailov, Kanishk Gandhi, Louis Castricato, Anikait Singh, Chase Blagden, Violet Xiang, Dakota Mahan, and Nick Haber. Big-math: A large-scale, high-quality math dataset for reinforcement learning in language models, 2025. URL https://arxiv.org/abs/2502.17387.

Mislav Balunović, Jasper Dekoninck, Ivo Petrov, Nikola Jovanović, and Martin Vechev. Matharena: Evaluating llms on uncontaminated math competitions, February 2025. URL https://matharena.ai/.

Shan Chen, Mingye Gao, Kuleen Sasse, Thomas Hartvigsen, Brian Anthony, Lizhou Fan, Hugo Aerts, Jack Gallifant, and Danielle S. Bitterman. When helpfulness backfires: Llms and the risk of false medical information due to sycophantic behavior. *npj Digital Medicine*, 8(1):605, Oct 2025. ISSN 2398-6352. doi: 10.1038/s41746-025-02008-z. URL https://doi.org/10.1038/s41746-025-02008-z.

DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

Will Epperson, Gagan Bansal, Victor C Dibia, Adam Fourney, Jack Gerrits, Erkang (Eric) Zhu, and Saleema Amershi. Interactive debugging and steering of multi-agent ai systems. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, CHI '25, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400713941. doi: 10.1145/3706598.3713581. URL https://doi.org/10.1145/3706598.3713581.

Aryo Pradipta Gema, Joshua Ong Jun Leang, Giwon Hong, Alessio Devoto, Alberto Carlo Maria Mancino, Rohit Saxena, Xuanli He, Yu Zhao, Xiaotang Du, Mohammad Reza Ghasemi Madani, Claire Barale, Robert McHardy, Joshua Harris, Jean Kaddour, Emile van Krieken, and Pasquale Minervini. Are we done with mmlu?, 2024.

Google. Gemini 2.5 pro model card. 2025a. URL https://storage.googleapis.com/deepmind-media/Model-Cards/Gemini-2-5-Pro-Model-Card.pdf.

Google. Gemini 3 pro model card. 2025b. URL https://storage.googleapis.com/deepmind-media/Model-Cards/Gemini-3-Pro-Model-Card.pdf.

Alex Gu, Baptiste Rozière, Hugh Leather, Armando Solar-Lezama, Gabriel Synnaeve, and Sida I. Wang. Cruxeval: A benchmark for code reasoning, understanding and execution. *arXiv preprint arXiv:2401.03065*, 2024.

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. URL https://openreview.net/forum?id=IkmD3fKBPQ.

Yiming Huang, Biquan Bie, Zuqiu Na, Weilin Ruan, Songxin Lei, Yutao Yue, and Xinlei He. An empirical study of the anchoring effect in llms: Existence, mechanism, and potential mitigations. *arXiv preprint arXiv:2505.15392*, 2025.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=chfJJYC3iL.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

Hynek Kydlíček. math-verify: A library for verifying mathematical answers. https://github.com/huggingface/Math-Verify, 2024.

Seongyun Lee, Yongrae Jo, Minju Seo, Moontae Lee, and Minjoon Seo. Lost in the noise: How reasoning models fail with contextual distractors. *arXiv preprint arXiv:2601.07226*, 2026.

Young-Jun Lee, Seungone Kim, Byung-Kwan Lee, Minkyeong Moon, Yechan Hwang, Jong Myoung Kim, Graham Neubig, Sean Welleck, and Ho-Jin Choi. Refinebench: Evaluating refinement capability of language models via checklists. *arXiv preprint arXiv:2511.22173*, 2025.

Wenzhe Li, Yong Lin, Mengzhou Xia, and Chi Jin. Rethinking mixture-of-agents: Is mixing different large language models beneficial? *arXiv preprint arXiv:2502.00674*, 2025.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024.

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

Lovish Madaan, Aniket Didolkar, Suchin Gururangan, John Quan, Ruan Silva, Ruslan Salakhutdinov, Manzil Zaheer, Sanjeev Arora, and Anirudh Goyal. Rethinking thinking tokens: Llms as improvement operators. *arXiv preprint arXiv:2510.01123*, 2025.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. Rethinking the role of demonstrations: What makes in-context learning work? In *Proceedings of the 2022 conference on empirical methods in natural language processing*, pp. 11048–11064, 2022.

Ivan Moshkov, Darragh Hanley, Ivan Sorokin, Shubham Toshniwal, Christof Henkel, Benedikt Schifferer, Wei Du, and Igor Gitman. Aimo-2 winning solution: Building state-of-the-art mathematical reasoning models with openmathreasoning dataset. *arXiv preprint arXiv:2504.16891*, 2025.

Jeremy K. Nguyen. Human bias in ai models? anchoring effects and mitigation strategies in large language models. *Journal of Behavioral and Experimental Finance*, 43:100971, 2024. ISSN 2214-6350. doi: https://doi.org/10.1016/j.jbef.2024.100971. URL https://www.sciencedirect.com/science/article/pii/S2214635024000868.

OpenAI. gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925*, 2025a.

OpenAI. Gpt-5 system card. 2025b. URL https://cdn.openai.com/gpt-5-system-card.pdf.

Daniel E O'Leary. An anchoring effect in large language models. *IEEE Intelligent Systems*, 40(2): 23–26, 2025.

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

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Mrinank Sharma, Meg Tong, Tomasz Korbak, David Duvenaud, Amanda Askell, Samuel R. Bowman, Esin DURMUS, Zac Hatfield-Dodds, Scott R Johnston, Shauna M Kravec, Timothy Maxwell, Sam McCandlish, Kamal Ndousse, Oliver Rausch, Nicholas Schiefer, Da Yan, Miranda Zhang, and Ethan Perez. Towards understanding sycophancy in language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=tvhaxkMKAn.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=vAElhFcKW6.

Adi Simhi, Fazl Barez, Martin Tutek, Yonatan Belinkov, and Shay B Cohen. Old habits die hard: How conversational history geometrically traps llms. *arXiv preprint arXiv:2603.03308*, 2026.

Kaya Stechly, Matthew Marquez, and Subbarao Kambhampati. GPT-4 doesn't know it's wrong: An analysis of iterative prompting for reasoning problems. In *NeurIPS 2023 Workshop on Robustness of Few-shot and Zero-shot Learning in Foundation Models (R0-FoMo)*, 2023. URL https://arxiv.org/abs/2310.12397.

Gemma Team. Gemma 4 model card. 2026. URL https://ai.google.dev/gemma/docs/core/model_card_4.

Miles Turpin, Julian Michael, Ethan Perez, and Samuel Bowman. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting. *Advances in Neural Information Processing Systems*, 36:74952–74965, 2023.

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. Association for Computational Linguistics, 2024.

Siddarth Venkatraman, Vineet Jain, Sarthak Mittal, Vedant Shah, Johan Obando-Ceron, Yoshua Bengio, Brian R Bartoldson, Bhavya Kailkhura, Guillaume Lajoie, Glen Berseth, et al. Recursive self-aggregation unlocks deep thinking in large language models. *arXiv preprint arXiv:2509.26626*, 2025.

Junlin Wang, Jue Wang, Ben Athiwaratkun, Ce Zhang, and James Zou. Mixture-of-agents enhances large language model capabilities. *arXiv preprint arXiv:2406.04692*, 2024.

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

Wenda Xu, Guanglei Zhu, Xuandong Zhao, Liangming Pan, Lei Li, and William Wang. Pride and prejudice: Llm amplifies self-bias in self-refinement. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15474–15492, 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.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. STaR: Bootstrapping reasoning with reasoning. In *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=_3ELRdg2sgI.

Kaizhong Zhang and Dennis Shasha. Simple fast algorithms for the editing distance between trees and related problems. *SIAM Journal on Computing*, 18(6):1245–1262, 1989. doi: 10.1137/0218082. URL https://doi.org/10.1137/0218082.

Muru Zhang, Ofir Press, William Merrill, Alisa Liu, and Noah A. Smith. How language model hallucinations can snowball. In *Proceedings of the 41st International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=FPlaQyAGHu.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: Llm agents are experiential learners. In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence and Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence and Fourteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'24/IAAI'24/EAAI'24. AAAI Press, 2024. ISBN 978-1-57735-887-9. doi: 10.1609/aaai.v38i17.29936. URL https://doi.org/10.1609/aaai.v38i17.29936.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. Llamafactory: Unified efficient fine-tuning of 100+ language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, Bangkok, Thailand, 2024. Association for Computational Linguistics. URL http://arxiv.org/abs/2403.13372.

Zhanke Zhou, Rong Tao, Jianing Zhu, Yiwen Luo, Zengmao Wang, and Bo Han. Can language models perform robust reasoning in chain-of-thought prompting with noisy rationales? *Advances in Neural Information Processing Systems*, 37:123846–123910, 2024.

Kunlun Zhu, Zijia Liu, Bingxuan Li, Muxin Tian, Yingxuan Yang, Jiaxun Zhang, Pengrui Han, Qipeng Xie, Fuyang Cui, Weijia Zhang, et al. Where llm agents fail and how they can learn from failures. *arXiv preprint arXiv:2509.25370*, 2025a.

Wang Bill Zhu, Tianqi Chen, Ching Ying Lin, Jade Law, Mazen Jizzini, Jorge J Nieva, Ruishan Liu, and Robin Jia. Cancer-myth: Evaluating ai chatbot on patient questions with false presuppositions. *arXiv preprint arXiv:2504.11373*, 2025b.

APPENDIX CONTENTS

A Additional Details of Contextual Drag Evaluation **16**

A.1 Prompt Templates 16
A.2 Inference Configurations 18
A.3 Anchor Models and Reasonably-Hard Subsets 18
A.4 Full Results of Contextual Drag Evaluation 19
A.5 Extension to Code Generation (LiveCodeBench v6) 22
A.6 Majority Voting and pass@k Baselines 22
A.7 Inference Cost of Recursive Filtering 22

B Unified Cross-Drag Setup and Results **24**

B.1 Anchor models and the unified draft-sampling setup 24
B.2 Cross-drag Pass@1 and Pass@5 25
B.3 Cross-drag on frontier proprietary models 25
B.4 Cross-drag structural similarity (TED) 28
B.5 Cross-drag under error signals 28
B.6 Cross-drag context denoising 28

C Ablation Evaluation of Contextual Drag **31**

C.1 Thinking Budget 31
C.2 Position and Verification 31
C.3 Matched-Prompt Control 33
C.4 Scaling: Varying the Numbers of In-Context Drafts with Mixed Correctness 34

D Tree-Edit Distance (TED) Analysis **36**

D.1 Implementation 36
D.2 Definition of 2F TED 36
D.3 Ablation of Answer Reuse Instruction 37
D.4 TED Analysis Under Error Signals 38

E Additional Analysis of Contextual Drag Despite Error Awareness **39**

F Additional Details of Context Denoising Experiments **42**

F.1 Implementation of REVISE and FILTER 42
F.2 Additional Results 44
F.3 Outcome-Bucket Decomposition of Mitigation 44
F.4 Recursive Filter: Per-Round Raw Statistics 44
F.5 Iterative Refinement and Recursive Filtering Beyond GPT-OSS-20B 48
F.6 Qualitative Case Study: FILTER vs. REVISE on the Same Draft 48
F.7 Relation to Prior Self-Improvement Strategies 49

G Conditional Fallback Training for Contextual Drag **50**

G.1 Overview 50
G.2 Data Curation: Question Selection from BigMathRL 50
G.3 Training Setup 52
G.4 Results 53
G.5 Comparison to Outcome-Based RL (GRPO) 54

A Additional Details of Contextual Drag Evaluation

Figure 7 illustrates the evaluation setup for contextual drag in Figure 2.

A.1 Prompt Templates

We provide the prompt templates used to evaluate contextual drag in Figure 2 below. We perform 2 types of inference:

Direct. given only the problem, the model is asked to solve the problem with chain-of-thoughts (Direct template);

Direct

```
{problem} Let's think step by step and please wrap your final answer in
\boxed{}.
```

1F/2F. given the problem and one or two incorrect model-generated draft solutions, the model is asked to check the correctness of the draft and generate a new solution (1F template, 2F template). We verify correctness by parsing the final answer and comparing the model's answer with the ground truth via a symbolic verifier Kydlíček (2024). We remove all the thinking traces wrapped in <think></think> from the drafts to keep the context length manageable. This also follows the documented default multi-turn behavior of the model families we evaluate: the GPT-OSS harmony format guide (OpenAI, 2025a), the Qwen3 model card and chat template (Yang et al., 2025), and the R1-Distill chat template (DeepSeek-AI, 2025) all drop the reasoning of earlier turns from the conversation history.

1F

```
In this task, you will be provided with a problem and one draft
solution to that problem.  For the draft, walk through the solution
step-by-step and check for any fatal mistakes in the reasoning (for
example:  incorrect use of theorems or facts, incorrect calculations,
etc.).  Please note that once you spot a fatal error in a draft's
reasoning, you **MUST** stop checking the remaining steps of that draft
and move on.

For the draft, output an overall verdict on the correctness of the
draft using the following tag format:

<overall_verdict>[Correct / Incorrect]</overall_verdict>

While evaluating individual steps, be strict about algebra, logic,
and correct theorem application.  If the draft is incorrect, write a
complete, correct solution (you may reuse sound ideas from the draft).
If the draft is correct, you may adopt and polish it as the final
solution.

Conclude by presenting the final answer only inside \boxed{} (i.e., the
final answer should appear only inside a LaTeX `\boxed{}`).

The problem is as follows:
-- beginning of problem --
{problem}
-- end of problem --

Here is the draft solution you need to consider:

### Solution:
-- beginning of the draft --
{draft1}
-- end of the draft --
```

```
Please:
1.  Carefully evaluate the draft solution step-by-step.  Once you spot
a fatal error in a draft's reasoning, you **MUST** stop checking the
remaining steps of that draft and move on.
2.  Output the correctness verdicts for the draft.
3.  Provide a correct solution to the problem with complete reasoning
steps that lead to the answer.

Remember:  conclude with the final answer only in \boxed{}.
```

2F

```
In this task, you will be provided with a problem and two draft
solutions to that problem.  For each draft, walk through the solution
step-by-step and check for any fatal mistakes in the reasoning (for
example:  incorrect use of theorems or facts, incorrect calculations,
etc.).  Please note that once you spot a fatal error in a draft's
reasoning, you **MUST** stop checking the remaining steps of that draft
and move on to the next draft.

For each draft (Solution 1 and Solution 2), output an overall verdict
on the correctness of the draft using the following tag format:

<overall_verdict_1>[Correct / Incorrect]</overall_verdict_1>
<overall_verdict_2>[Correct / Incorrect]</overall_verdict_2>

While evaluating individual steps, be strict about algebra, logic, and
correct theorem application.  If both drafts are incorrect, write
a complete, correct solution (you may reuse sound ideas from the
drafts).  If a draft is correct, you may adopt and polish it as the
final solution.

Conclude by presenting the final answer only inside \boxed{} (i.e., the
final answer should appear only inside a LaTeX '\boxed{}').

The problem is as follows:
-- beginning of problem --
{problem}
-- end of problem --

Here are the two draft solutions you need to consider:

### Solution 1:
-- beginning of the first draft --
{draft1}
-- end of the first draft --

### Solution 2:
-- beginning of the second draft --
{draft2}
-- end of the second draft --

Please:
1.  Carefully evaluate the two draft solutions step-by-step.  Once you
spot a fatal error in a draft's reasoning, you **MUST** stop checking
the remaining steps of that draft and move on to the next draft.
2.  Output the correctness verdicts for the drafts, respectively.
3.  Provide a correct solution to the problem with complete reasoning
steps that lead to the answer.

Remember:  conclude with the final answer only in \boxed{}.
```

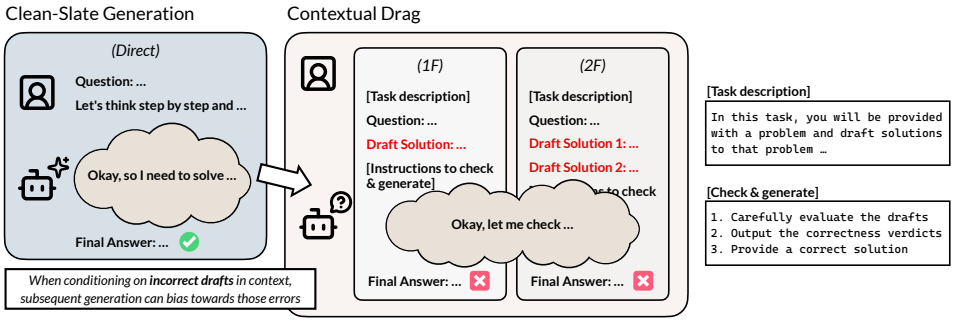

Figure 7: **Contextual drag evaluation setup**: We measure (1) DIRECT performance of the model's clean-slate generation, and (2) 1F/2F performance of the model's generation conditioned on 1 or 2 incorrect draft solutions. DIRECT evaluation uses the standard chain-of-thought prompting. For 1F/2F evaluations, the model is instructed to perform a step-by-step verification then produce a correctness verdict for each in-context draft, then produce a new solution.

A.2 INFERENCE CONFIGURATIONS

We followed the official recommended generation configurations for model inference, as listed in Table 1. For efficient inference, we use vLLM project version 0.10.2 Kwon et al. (2023) for all models except for Gemma 4. For Gemma 4 we use vllM project version 0.19.2. We use a total context size of 32,768 for all models except for the Nemotron family, which has a recommended max output token limit of 65,536. We run inference on 4-8 NVIDIA H100 GPUs.

Table 1: Inference configurations for all evaluated models. We report the maximum context length and the sampling parameters used during generation. For models where `top_k` sampling is not being used, we denote it as "–".

Model	#tokens	Temp	Top-p / Top-k	Huggingface Path
Qwen3-8B (Thinking)	32,768	0.6	0.95 / 20	`Qwen/Qwen3-8B`
Qwen3-32B (Thinking)	32,768	0.6	0.95 / 20	`Qwen/Qwen3-32B`
Nemotron-7B	65,536	0.6	0.95 / –	`nvidia/OpenReasoning-Nemotron-7B`
Nemotron-32B	65,536	0.6	0.95 / –	`nvidia/OpenReasoning-Nemotron-32B`
GPT-OSS-20B	32,768	1.0	1.00 / 40	`openai/gpt-oss-20b`
GPT-OSS-120B	32,768	1.0	1.00 / 40	`openai/gpt-oss-120b`
LlamaR1-8B	32,768	0.6	0.95 / –	`deepseek-ai/DeepSeek-R1-Distill-Llama-8B`
QwenR1-7B	32,768	0.6	0.95 / –	`deepseek-ai/DeepSeek-R1-Distill-Qwen-7B`
Gemma4-31B	32,768	1.0	0.95 / 64	`google/gemma-4-31B-it`
Gemma4-E4B	32,768	1.0	0.95 / 64	`google/gemma-4-E4B-it`

A.3 ANCHOR MODELS AND REASONABLY-HARD SUBSETS

The selection of anchor models, the per-task draft pool, and the reasonably-hard subset together form the unified evaluation setup underlying both self-drag (this section) and cross-drag (Section B). Self-drag is the special case in which the evaluated model is itself one of that task's anchor models, so its in-context drafts are drawn from its own clean-slate generations; cross-drag uses drafts from the remaining anchors. We document the full setup (including the clean-slate initial-sampling results, Table 8, and the per-task anchor lineups, Table 9) in Section B.1.

The size of each reasonably-hard subset (the number of problems retained after the per-(model, task) ≥ 2 correct / ≥ 2 incorrect filter) is reported in Table 2 for both self-drag (per-model rows) and cross-drag (shared *Cross subset* row).

Table 2: Number of problems selected for each model (**#problems** per task–model cell) used in self-drag and cross-drag evaluation. The *Original* row is the full task pool. The *Cross subset* row is the shared cross-drag pool (where curation is explained in more details in Table 8). Each remaining row is a per-model self-drag evaluation subset (unique problem ids in the 1F-conditioned evaluation, identical for Direct/1F/2F).

	AIME24	AIME25	HMMT24	HMMT25	GPQA-D	MMLU	CRUX-I	24Game
Original	30	30	30	30	198	2629	800	1362
Cross subset	10	13	16	21	132	569	211	653
GPT-OSS-120B	8	9	14	17	92	301	369	520
GPT-OSS-20B	10	14	17	17	133	551	708	563
Nemotron-32B	3	3	10	7	80	305	102	311
Nemotron-7B	5	7	12	14	106	593	712	405
Qwen3-32B	8	11	13	11	72	232	108	391
Qwen3-8B	6	12	6	8	73	287	109	351
LlamaR1-8B	19	9	8	6	137	1496	497	789
QwenR1-7B	16	13	8	12	76	1489	412	98
Gemma4-31B	4	5	12	7	41	100	14	12
Gemma4-E4B	14	12	9	10	105	492	337	182

A.4 Full Results of Contextual Drag Evaluation

Table 3 and Table 4 present the full pass@1 and pass@5 self-drag evaluation results that summarize Figure 2. Drafts are sampled from each model's own clean-slate generations; we conduct 16 independent inferences per problem and report deltas relative to DIRECT alongside 95% confidence intervals across trajectories. The cross-drag analogues (drafts from other anchor models, including proprietary baselines) are reported in Section B.

We note that there are no substantial differences between the pass@1 and pass@5 results, demonstrating that contextual drag leads to strong, consistent performance degradation that is hard to recover from by increasing the number of independent samplings.

Table 3: **Full pass@1 results for self-drag** (mean ± 95% confidence interval, in pp; Bernoulli normal-approximation with $n = n_{\text{problems}} \times n_{\text{trajectories}}$). Drafts in context are sampled from the model's own clean-slate (DIRECT) generations; we report performance under one and two in-context drafts (1F and 2F). Subscripted deltas are absolute pp differences relative to DIRECT. DIRECT is the selection-coupled estimate, computed on the same 16 samples used to select the subset.

	AIME24*			AIME25*		
	DIRECT	1F	2F	DIRECT	1F	2F
GPT-OSS-120B	64.06 ±8.31	29.69 ±7.92 (-34.38)	33.59 ±8.18 (-30.47)	63.19 ±7.88	57.64 ±8.07 (-5.56)	44.44 ±8.12 (-18.75)
GPT-OSS-20B	54.37 ±7.72	26.25 ±6.82 (-28.12)	21.88 ±6.41 (-32.50)	66.96 ±6.16	24.11 ±5.60 (-42.86)	34.82 ±6.24 (-32.14)
Nemotron-32B	68.75 ±13.11	45.83 ±14.10 (-22.92)	45.83 ±14.10 (-22.92)	56.25 ±14.03	35.42 ±13.53 (-20.83)	45.83 ±14.10 (-10.42)
Nemotron-7B	56.25 ±10.87	36.25 ±10.53 (-20.00)	36.25 ±10.53 (-20.00)	58.04 ±9.14	37.50 ±8.97 (-20.54)	33.93 ±8.77 (-24.11)
Qwen3-32B	58.59 ±8.53	28.91 ±7.85 (-29.69)	28.91 ±7.85 (-29.69)	70.45 ±6.74	30.68 ±6.81 (-39.77)	27.27 ±6.58 (-43.18)
Qwen3-8B	56.25 ±9.92	23.96 ±8.54 (-32.29)	19.79 ±7.97 (-36.46)	68.75 ±6.56	41.67 ±6.97 (-27.08)	31.77 ±6.59 (-36.98)
LlamaR1-8B	47.70 ±5.61	31.25 ±5.21 (-16.45)	21.71 ±4.63 (-25.99)	50.00 ±8.17	20.14 ±6.55 (-29.86)	18.75 ±6.38 (-31.25)
QwenR1-7B	52.34 ±6.12	30.47 ±5.64 (-21.88)	22.27 ±5.10 (-30.08)	44.23 ±6.75	30.77 ±6.27 (-13.46)	19.71 ±5.41 (-24.52)
Gemma4-31B	62.50 ±11.86	43.75 ±12.15 (-18.75)	45.31 ±12.20 (-17.19)	66.25 ±10.36	63.75 ±10.53 (-2.50)	57.50 ±10.83 (-8.75)
Gemma4-E4B	56.70 ±6.49	35.27 ±6.26 (-21.43)	33.48 ±6.18 (-23.21)	56.25 ±7.02	39.06 ±6.90 (-17.19)	50.00 ±7.07 (-6.25)
	HMMT24*			HMMT25*		
	DIRECT	1F	2F	DIRECT	1F	2F
GPT-OSS-120B	48.66 ±6.55	33.04 ±6.16 (-15.62)	41.07 ±6.44 (-7.59)	58.09 ±5.86	40.44 ±5.83 (-17.65)	40.07 ±5.82 (-18.01)
GPT-OSS-20B	47.43 ±5.93	19.12 ±4.67 (-28.31)	22.43 ±4.96 (-25.00)	55.51 ±5.91	17.65 ±4.53 (-37.87)	19.12 ±4.67 (-36.40)
Nemotron-32B	56.88 ±7.67	41.88 ±7.64 (-15.00)	35.62 ±7.42 (-21.25)	66.96 ±8.71	46.43 ±9.24 (-20.54)	44.64 ±9.21 (-22.32)
Nemotron-7B	40.62 ±6.95	33.85 ±6.69 (-6.77)	34.38 ±6.72 (-6.25)	52.23 ±6.54	41.07 ±6.44 (-11.16)	36.61 ±6.31 (-15.63)
Qwen3-32B	48.08 ±6.79	23.56 ±5.77 (-24.52)	18.27 ±5.25 (-29.81)	48.86 ±7.39	13.07 ±4.98 (-35.80)	11.36 ±4.69 (-37.50)
Qwen3-8B	34.37 ±9.50	33.33 ±9.43 (-1.04)	33.33 ±9.43 (-1.04)	46.88 ±8.65	24.22 ±7.42 (-22.66)	12.50 ±5.73 (-34.38)
LlamaR1-8B	51.56 ±8.66	25.78 ±7.58 (-25.78)	17.97 ±6.65 (-33.59)	48.96 ±10.00	25.00 ±8.66 (-23.96)	28.12 ±8.99 (-20.83)
QwenR1-7B	37.50 ±8.39	21.88 ±7.16 (-15.62)	24.22 ±7.42 (-13.28)	32.81 ±6.64	13.54 ±4.84 (-19.27)	11.46 ±4.51 (-21.35)
Gemma4-31B	55.73 ±7.03	59.38 ±6.95 (+3.65)	50.52 ±7.07 (-5.21)	69.64 ±8.52	62.50 ±8.97 (-7.14)	50.00 ±9.26 (-19.64)
Gemma4-E4B	34.03 ±7.74	22.22 ±6.79 (-11.81)	10.42 ±4.99 (-23.61)	51.87 ±7.74	38.12 ±7.53 (-13.75)	30.00 ±7.10 (-21.87)
	GPQA-DIAMOND*			MMLU-REDUX*		
	DIRECT	1F	2F	DIRECT	1F	2F
GPT-OSS-120B	57.47 ±2.53	31.66 ±2.38 (-25.82)	29.76 ±2.34 (-27.72)	56.89 ±1.40	28.92 ±1.28 (-27.97)	20.85 ±1.15 (-36.05)
GPT-OSS-20B	55.03 ±2.11	23.68 ±1.81 (-31.34)	25.80 ±1.86 (-29.23)	59.89 ±1.02	26.70 ±0.92 (-33.19)	21.67 ±0.86 (-38.23)
Nemotron-32B	55.94 ±2.72	24.84 ±2.37 (-31.09)	17.73 ±2.09 (-38.20)	56.91 ±1.39	13.34 ±0.95 (-43.57)	4.24 ±0.57 (-52.66)
Nemotron-7B	50.06 ±2.38	20.58 ±1.92 (-29.48)	11.38 ±1.51 (-38.68)	57.19 ±1.00	13.64 ±0.69 (-43.55)	7.09 ±0.52 (-50.09)
Qwen3-32B	50.26 ±2.89	19.27 ±2.28 (-30.99)	7.12 ±1.48 (-43.14)	56.25 ±1.60	23.03 ±1.35 (-33.22)	6.87 ±0.81 (-49.38)
Qwen3-8B	56.08 ±2.85	16.87 ±2.15 (-39.21)	10.10 ±1.73 (-45.98)	56.84 ±1.43	14.00 ±1.00 (-42.84)	6.12 ±0.69 (-50.72)
LlamaR1-8B	46.85 ±2.09	23.27 ±1.77 (-23.59)	20.03 ±1.68 (-26.82)	58.35 ±0.62	44.82 ±0.63 (-13.53)	42.26 ±0.63 (-16.09)
QwenR1-7B	43.83 ±2.79	17.60 ±2.14 (-26.23)	18.34 ±2.18 (-25.49)	54.05 ±0.63	44.15 ±0.63 (-9.90)	47.82 ±0.63 (-6.23)
Gemma4-31B	54.88 ±3.81	35.67 ±3.67 (-19.21)	24.70 ±3.30 (-30.18)	54.75 ±2.44	26.00 ±2.15 (-28.75)	17.38 ±1.86 (-37.38)
Gemma4-E4B	50.18 ±2.39	9.76 ±1.42 (-40.42)	7.20 ±1.24 (-42.98)	56.15 ±1.10	10.81 ±0.69 (-45.34)	6.17 ±0.53 (-49.97)
	CRUXEVAL-I*			GAME OF 24*		
	DIRECT	1F	2F	DIRECT	1F	2F
GPT-OSS-120B	81.40 ±0.99	93.41 ±0.75 (+12.01)	91.24 ±0.72 (+9.84)	67.14 ±1.01	49.82 ±1.07 (-17.32)	45.20 ±1.07 (-21.94)
GPT-OSS-20B	70.37 ±0.84	78.24 ±0.79 (+7.87)	75.99 ±0.79 (+5.61)	70.97 ±0.94	32.79 ±0.97 (-38.18)	26.78 ±0.91 (-44.19)
Nemotron-32B	72.55 ±2.17	40.69 ±2.38 (-31.86)	43.75 ±2.41 (-28.80)	66.18 ±1.31	47.43 ±1.39 (-18.75)	42.06 ±1.37 (-24.12)
Nemotron-7B	55.15 ±0.91	12.55 ±0.61 (-42.60)	10.41 ±0.56 (-44.74)	65.31 ±1.16	43.61 ±1.21 (-21.70)	35.32 ±1.16 (-29.98)
Qwen3-32B	66.49 ±2.23	40.39 ±2.35 (-26.10)	33.56 ±2.23 (-32.93)	68.48 ±1.15	32.82 ±1.16 (-35.66)	15.54 ±0.90 (-52.94)
Qwen3-8B	62.84 ±2.27	30.22 ±2.16 (-32.63)	26.49 ±2.07 (-36.35)	64.67 ±1.25	20.53 ±1.06 (-44.14)	16.60 ±0.97 (-48.08)
LlamaR1-8B	52.35 ±1.10	39.95 ±1.08 (-12.40)	37.46 ±1.06 (-14.89)	48.05 ±0.87	9.09 ±0.50 (-38.96)	3.30 ±0.31 (-44.75)
QwenR1-7B	57.28 ±1.19	27.00 ±1.07 (-30.28)	21.48 ±0.99 (-35.80)	38.46 ±2.41	15.82 ±1.81 (-22.64)	14.35 ±1.74 (-24.11)
Gemma4-31B	73.21 ±5.80	43.75 ±6.50 (-29.46)	45.54 ±6.52 (-27.68)	61.46 ±6.88	47.40 ±7.06 (-14.06)	42.19 ±6.99 (-19.27)
Gemma4-E4B	63.76 ±1.28	17.86 ±1.02 (-45.90)	11.59 ±0.85 (-52.17)	66.21 ±1.72	36.09 ±1.74 (-30.12)	22.01 ±1.50 (-44.20)

Table 4: **Full pass@5 results for self-drag** (mean $\pm$ 95% confidence interval, in pp; Bernoulli normal-approximation with $n = n_{\text{problems}} \times n_{\text{trajectories}}$). Drafts in context are sampled from the model's own clean-slate (DIRECT) generations; we report performance under one and two in-context drafts (1F and 2F). Subscripted deltas are absolute pp differences relative to DIRECT.

	AIME24*			AIME25*		
	DIRECT	1F	2F	DIRECT	1F	2F
GPT-OSS-120B	96.88 ±3.01	76.56 ±7.34 (-20.31)	71.09 ±7.85 (-25.78)	96.53 ±2.99	92.36 ±4.34 (-4.17)	70.83 ±7.42 (-25.69)
GPT-OSS-20B	93.12 ±3.92	50.62 ±7.75 (-42.50)	55.62 ±7.70 (-37.50)	94.64 ±2.95	54.91 ±6.52 (-39.73)	72.32 ±5.86 (-22.32)
Nemotron-32B	100.00 ±0.00	66.67 ±13.34 (-33.33)	85.42 ±9.98 (-14.58)	97.92 ±4.04	56.25 ±14.03 (-41.67)	83.33 ±10.54 (-14.58)
Nemotron-7B	98.75 ±2.43	67.50 ±10.26 (-31.25)	73.75 ±9.64 (-25.00)	91.07 ±5.28	85.71 ±6.48 (-5.36)	66.07 ±8.77 (-25.00)
Qwen3-32B	98.44 ±2.15	58.59 ±8.53 (-39.84)	61.72 ±8.42 (-36.72)	100.00 ±0.00	78.41 ±6.08 (-21.59)	60.23 ±7.23 (-39.77)
Qwen3-8B	89.58 ±6.11	42.71 ±9.90 (-46.88)	31.25 ±9.27 (-58.33)	96.88 ±2.46	72.92 ±6.29 (-23.96)	59.38 ±6.95 (-37.50)
LlamaR1-8B	90.46 ±3.30	67.11 ±5.28 (-23.36)	57.24 ±5.56 (-33.22)	96.53 ±2.99	68.06 ±7.62 (-28.47)	56.94 ±8.09 (-39.58)
QwenR1-7B	95.31 ±2.59	66.41 ±5.79 (-28.91)	54.30 ±6.10 (-41.02)	87.50 ±4.49	61.54 ±6.61 (-25.96)	62.02 ±6.60 (-25.48)
Gemma4-31B	96.88 ±4.26	84.38 ±8.90 (-12.50)	64.06 ±11.76 (-32.81)	100.00 ±0.00	100.00 ±0.00 (+0.00)	97.50 ±3.42 (-2.50)
Gemma4-E4B	93.75 ±3.17	55.80 ±6.50 (-37.95)	44.20 ±6.50 (-49.55)	93.23 ±3.55	59.38 ±6.95 (-33.85)	69.27 ±6.53 (-23.96)
	HMMT24*			HMMT25*		
	DIRECT	1F	2F	DIRECT	1F	2F
GPT-OSS-120B	90.18 ±3.90	70.98 ±5.94 (-19.20)	79.02 ±5.33 (-11.16)	88.24 ±3.83	69.85 ±5.45 (-18.38)	66.54 ±5.61 (-21.69)
GPT-OSS-20B	89.71 ±3.61	50.74 ±5.94 (-38.97)	48.53 ±5.94 (-41.18)	88.60 ±3.78	42.65 ±5.88 (-45.96)	43.75 ±5.90 (-44.85)
Nemotron-32B	95.00 ±3.38	80.62 ±6.12 (-14.38)	76.88 ±6.53 (-18.12)	98.21 ±2.45	77.68 ±7.71 (-20.54)	73.21 ±8.20 (-25.00)
Nemotron-7B	88.02 ±4.59	63.02 ±6.83 (-25.00)	67.19 ±6.64 (-20.83)	94.20 ±3.06	72.77 ±5.83 (-21.43)	58.48 ±6.45 (-35.71)
Qwen3-32B	86.06 ±4.71	47.60 ±6.79 (-38.46)	36.54 ±6.54 (-49.52)	85.23 ±5.24	29.55 ±6.74 (-55.68)	28.98 ±6.70 (-56.25)
Qwen3-8B	88.54 ±6.37	57.29 ±9.90 (-31.25)	65.62 ±9.50 (-22.92)	86.72 ±5.88	35.16 ±8.27 (-51.56)	35.94 ±8.31 (-50.78)
LlamaR1-8B	92.97 ±4.43	63.28 ±8.35 (-29.69)	55.47 ±8.61 (-37.50)	92.71 ±5.20	52.08 ±9.99 (-40.63)	54.17 ±9.97 (-38.54)
QwenR1-7B	72.66 ±7.72	42.19 ±8.56 (-30.47)	39.84 ±8.48 (-32.81)	77.60 ±5.90	33.85 ±6.69 (-43.75)	34.90 ±6.74 (-42.71)
Gemma4-31B	89.58 ±4.32	87.50 ±4.68 (-2.08)	87.50 ±4.68 (-2.08)	100.00 ±0.00	99.11 ±1.74 (-0.89)	89.29 ±5.73 (-10.71)
Gemma4-E4B	84.72 ±5.88	66.67 ±7.70 (-18.06)	33.33 ±7.70 (-51.39)	95.00 ±3.38	62.50 ±7.50 (-32.50)	54.37 ±7.72 (-40.63)
	GPQA-DIAMOND*			MMLU-REDUX*		
	DIRECT	1F	2F	DIRECT	1F	2F
GPT-OSS-120B	92.39 ±1.35	60.80 ±2.49 (-31.59)	59.44 ±2.51 (-32.95)	92.55 ±0.74	54.65 ±1.41 (-37.89)	42.36 ±1.40 (-50.19)
GPT-OSS-20B	92.06 ±1.15	56.77 ±2.10 (-35.29)	57.99 ±2.10 (-34.07)	94.59 ±0.47	56.01 ±1.04 (-38.58)	47.90 ±1.04 (-46.69)
Nemotron-32B	92.89 ±1.41	54.22 ±2.73 (-38.67)	37.66 ±2.65 (-55.23)	93.14 ±0.71	29.61 ±1.28 (-63.52)	11.17 ±0.88 (-81.97)
Nemotron-7B	88.62 ±1.51	46.88 ±2.37 (-41.75)	30.90 ±2.20 (-57.72)	92.91 ±0.52	31.59 ±0.94 (-61.32)	17.01 ±0.76 (-75.90)
Qwen3-32B	89.32 ±1.78	44.88 ±2.87 (-44.44)	19.10 ±2.27 (-70.23)	92.89 ±0.83	46.07 ±1.60 (-46.82)	18.18 ±1.24 (-74.70)
Qwen3-8B	91.35 ±1.61	39.13 ±2.80 (-52.23)	22.52 ±2.40 (-68.84)	92.66 ±0.75	30.27 ±1.33 (-62.39)	15.05 ±1.03 (-77.61)
LlamaR1-8B	87.59 ±1.38	49.50 ±2.09 (-38.09)	38.96 ±2.04 (-48.63)	93.66 ±0.31	67.27 ±0.59 (-26.40)	60.56 ±0.62 (-33.10)
QwenR1-7B	87.01 ±1.89	43.01 ±2.78 (-44.00)	43.42 ±2.79 (-43.59)	92.33 ±0.34	69.17 ±0.59 (-23.16)	67.29 ±0.60 (-25.04)
Gemma4-31B	94.05 ±1.81	63.87 ±3.68 (-30.18)	46.19 ±3.82 (-47.87)	92.75 ±1.27	46.44 ±2.44 (-46.31)	30.94 ±2.26 (-61.81)
Gemma4-E4B	90.30 ±1.42	22.20 ±1.99 (-68.10)	17.32 ±1.81 (-72.98)	92.06 ±0.60	22.14 ±0.92 (-69.92)	12.26 ±0.72 (-79.81)
	CRUXEVAL-I*			GAME OF 24*		
	DIRECT	1F	2F	DIRECT	1F	2F
GPT-OSS-120B	99.83 ±0.10	97.89 ±0.43 (-1.95)	98.53 ±0.31 (-1.30)	97.20 ±0.35	77.06 ±0.90 (-20.14)	75.25 ±0.93 (-21.95)
GPT-OSS-20B	98.91 ±0.19	95.01 ±0.42 (-3.90)	96.03 ±0.36 (-2.88)	98.77 ±0.23	72.50 ±0.92 (-26.27)	65.70 ±0.98 (-33.07)
Nemotron-32B	95.65 ±0.99	67.46 ±2.27 (-28.19)	70.89 ±2.20 (-24.75)	96.52 ±0.51	84.04 ±1.02 (-12.48)	79.94 ±1.11 (-16.58)
Nemotron-7B	94.38 ±0.42	24.17 ±0.79 (-70.21)	30.07 ±0.84 (-64.31)	96.59 ±0.44	83.92 ±0.89 (-12.67)	74.12 ±1.07 (-22.47)
Qwen3-32B	96.18 ±0.90	68.69 ±2.22 (-27.49)	64.35 ±2.26 (-31.83)	97.06 ±0.42	69.07 ±1.15 (-27.99)	41.22 ±1.22 (-55.83)
Qwen3-8B	96.50 ±0.86	58.14 ±2.32 (-38.36)	52.29 ±2.34 (-44.21)	95.74 ±0.53	46.08 ±1.30 (-49.66)	40.71 ±1.28 (-55.04)
LlamaR1-8B	91.64 ±0.61	67.34 ±1.03 (-24.30)	63.15 ±1.06 (-28.48)	90.30 ±0.52	26.03 ±0.77 (-64.27)	10.77 ±0.54 (-79.53)
QwenR1-7B	93.10 ±0.61	56.63 ±1.20 (-36.47)	49.70 ±1.21 (-43.40)	73.02 ±2.20	38.97 ±2.41 (-34.06)	31.82 ±2.31 (-41.20)
Gemma4-31B	97.77 ±1.93	69.20 ±6.05 (-28.57)	83.04 ±4.92 (-14.73)	95.31 ±2.99	79.69 ±5.69 (-15.62)	70.31 ±6.46 (-25.00)
Gemma4-E4B	97.44 ±0.42	42.08 ±1.32 (-55.36)	28.82 ±1.21 (-68.62)	95.50 ±0.75	75.69 ±1.56 (-19.81)	56.80 ±1.80 (-38.70)

A.5 Extension to Code Generation (LiveCodeBench v6)

We apply the same self-drag protocol to code generation on LiveCodeBench v6 (Jain et al., 2025), using a random sample of 200 problems and the same ≥ 2 correct / ≥ 2 incorrect filter. Table 5 reports clean-slate pass@1 on all 200 problems, the number of retained problems, and DIRECT and 1F pass@1 on the retained subset. A single incorrect self-draft lowers pass@1 for all three models, by 10.06 to 41.25 points.

Table 5: **Self-drag on LiveCodeBench v6** (pass@1, %). "All 200" is clean-slate pass@1 on the full 200-problem sample; n is the number of problems retained by the reasonably-hard filter; DIRECT, 1F and Δ are computed on the retained subset.

Model	All 200	n	DIRECT	1F	Δ
GPT-OSS-120B	73.00	60	58.23	16.98	-41.25
GPT-OSS-20B	65.31	78	55.93	22.20	-33.73
Gemma4-31B	80.91	41	61.74	51.68	-10.06

A.6 Majority Voting and pass@k Baselines

A natural alternative to reusing prior attempts is to discard them and aggregate independent samples, as in self-consistency (Wang et al., 2023). Table 6 reports majority voting (maj@k) and pass@k for $k \in \{1, 4, 8, 16\}$ under DIRECT, 1F, and 2F. Voting does not rescue error-conditioned generation; it widens the gap. From $k = 1$ to $k = 16$, majority voting improves DIRECT by 17.6 points (selection-coupled; see the caption of Table 6) but 1F by only 6.1 and 2F by 0.8. This is consistent with the structural analysis in Section 2.3: dragged samples make correlated errors biased toward the in-context draft rather than independent errors, so the vote consolidates on the dragged answer. pass@k shows the same ordering.

Table 6: **Majority voting and pass@k under contextual drag** (%, unweighted mean over 40 (model, task) cells: 10 models $\times$ AIME25*, HMMT25*, GPQA-DIAMOND* and MMLU-REDUX*). DIRECT pass@16 is 100% by construction of the reasonably-hard subset (Section A.3). More generally, the DIRECT columns are computed on the same 16 samples used to define the subset, so the DIRECT pass@k values should be read as upper bounds rather than independent estimates.

	maj@k			pass@k		
k	DIRECT	1F	2F	DIRECT	1F	2F
1	55.51	29.41	25.01	55.51	29.41	25.01
4	65.12	30.22	25.59	89.38	52.43	45.33
8	70.55	30.66	25.58	97.36	62.93	54.95
16	73.07	35.55	25.82	100.00	76.07	62.48
Gain 1 $\rightarrow$ 16	+17.56	+6.13	+0.80	+44.50	+46.66	+37.47

A.7 Inference Cost of Recursive Filtering

We report the per-round wall-clock breakdown of the recursive filtering pipeline used in Section 4.1 (Figure 3). The strategy step runs once per task, and each refinement round adds one filter call ($\sim$26–29 s) on top of one solve call ($\sim$60–97 s)—a constant per-round overhead of roughly one third of solve time, while flipping deterioration into improvement.

Table 7: **Per-stage inference time for one round of recursive FILTER.** Wall-clock cost of the recursive FILTER pipeline that converts self-deterioration into self-improvement in Figure 3 (GPT-OSS-20B). Strategy generation runs once per task, and each refinement round adds one filter call ($\sim$26–29 s) on top of one solve call ($\sim$60–97 s)—an overhead roughly a third of the per-round solve cost. The 16 trajectories per task are sampled to estimate accuracy, not as a deployment pattern.

Stage	AIME24	AIME25	HMMT24	HMMT25
Strategy Generation	40.1s	56.1s	40.0s	40.0s
Filtering (avg. across 15 runs)	28.8s	28.5s	25.7s	26.0s
Solving (avg. across 15 runs)	60.2s	71.3s	87.5s	96.7s

B UNIFIED CROSS-DRAG SETUP AND RESULTS

The main text (Section 2) focuses on *self-drag*, the special case in which the evaluated model is conditioned on its own incorrect drafts. This appendix documents the more general setup from which both self-drag and cross-drag are instantiated, and reports the cross-drag analogues of every analysis in Sections 2, 3 and 4.1, including results for frontier proprietary models that are infeasible to evaluate at full self-drag scale.

B.1 ANCHOR MODELS AND THE UNIFIED DRAFT-SAMPLING SETUP

The fundamental setup underlying both self-drag and cross-drag is to fix a small, high-quality *anchor pool* per task and to draw all in-context drafts from that pool. This avoids the well-known confound where conditioning on trivially weak drafts trivially degrades performance (Li et al., 2025).

Anchor models. For each task, we run clean-slate inference on the full open-weight lineup (Section 2.1, 16 samples per problem) and pick the top-3 models by clean-slate pass@1 as that task's *anchor models*. The corresponding clean-slate generations form the per-task draft pool. Table 8 reports the clean-slate accuracies that drive this selection, and Table 9 lists the resulting anchor lineups.

Table 8: **Initial evaluation results of open-weight models on all evaluation benchmarks**: We first run clean-slate generation for the set of open-weight models on the original data of all benchmarks to select the top-3 performant models (underlined), whose responses are used as the in-context drafts for later evaluation of contextual drag.

Model	AIME24	AIME25	HMMT24	HMMT25
GPT-OSS-120B	79.79 (±0.45)	78.54 (±0.41)	52.92 (±0.48)	56.04 (±0.41)
GPT-OSS-20B	72.71 (±0.46)	68.33 (±0.60)	46.88 (±0.41)	54.37 (±0.46)
Nemotron-32B	87.92 (±0.35)	83.75 (±0.35)	59.79 (±0.54)	69.79 (±0.34)
Nemotron-7B	81.88 (±0.30)	79.17 (±0.36)	52.71 (±0.49)	57.50 (±0.55)
Qwen3-32B	80.00 (±0.33)	71.67 (±0.45)	47.29 (±0.51)	48.12 (±0.48)
Qwen3-8B	75.00 (±0.28)	67.08 (±0.39)	38.12 (±0.28)	42.92 (±0.39)
LlamaR1-8B	43.33 (±0.71)	32.29 (±0.42)	23.54 (±0.31)	20.21 (±0.39)
QwenR1-7B	54.17 (±0.71)	36.25 (±0.48)	28.96 (±0.46)	26.67 (±0.47)
Model	GPQA-DIAMOND	MMLU-REDUX	CRUXEVAL-I	GAME OF 24
GPT-OSS-120B	69.35 (±0.07)	90.90 (±0.00)	89.51 (±0.07)	85.95 (±0.01)
GPT-OSS-20B	58.59 (±0.08)	87.05 (±0.00)	71.97 (±0.21)	86.28 (±0.02)
Nemotron-32B	75.19 (±0.04)	90.87 (±0.00)	93.27 (±0.01)	89.96 (±0.01)
Nemotron-7B	62.63 (±0.07)	84.24 (±0.00)	55.13 (±0.11)	87.02 (±0.01)
Qwen3-32B	69.16 (±0.06)	91.64 (±0.00)	94.11 (±0.01)	89.15 (±0.01)
Qwen3-8B	60.64 (±0.07)	88.66 (±0.00)	90.91 (±0.01)	88.11 (±0.01)
LlamaR1-8B	42.87 (±0.08)	66.72 (±0.01)	56.16 (±0.02)	52.41 (±0.02)
QwenR1-7B	34.69 (±0.04)	61.19 (±0.01)	67.57 (±0.02)	75.19 (±0.01)

Cross-drag setup. For each (task, evaluated model) pair, in-context drafts are drawn from the task's anchor pool, with any drafts produced by the evaluated model itself excluded so that the conditioning is strictly cross-model. This corresponds to multi-agent aggregation, draft handoff, and the frontier-model setup where running 16 self-drafts per problem is infeasible. The conditioning setup, the 1F/2F prompt templates (Section A.1), and the reasonably-hard subset (next paragraph) are identical to the self-drag configuration documented in Section A; only the source of the in-context drafts differs.

Reasonably-hard subset. We restrict each task to problems on which the anchor pool exhibits *both* successes and failures: each retained problem has at least two correct and at least two incorrect clean-slate responses across the anchors. This filter removes saturated and unsolved instances and isolates the regime where test-time scaling actually matters: problems for which pass@k exceeds

Table 9: List of anchor models whose clean-slate responses are used as in-context drafts in the evaluation of contextual drag in Figure 2.

Task	Anchor Models
AIME24	Nemotron-7B/32B, Qwen3-32B
AIME25	GPT-OSS-120B, Nemotron-7B/32B
HMMT24	GPT-OSS-120B, Nemotron-7B/32B
HMMT25	GPT-OSS-120B, Nemotron-7B/32B
GPQA-DIAMOND	GPT-OSS-120B, Nemotron-32B, Qwen3-32B
MMLU-REDUX	GPT-OSS-120B, Nemotron-32B, Qwen3-32B
CRUXEVAL-I	Nemotron-32B, Qwen3-8B/32B
GAME OF 24	Nemotron-32B, Qwen3-8B/32B

pass@1. Filtered subset sizes are reported in Table 2. Because subsets are defined from the anchor pool, they are shared between self-drag and cross-drag for a given task, making the two evaluations directly comparable.

B.2 CROSS-DRAG PASS@1 AND PASS@5

Figure 8 summarizes cross-drag pass@1 as Δ relative to DIRECT, and Tables 10 and 11 report the per-task numbers with 95% confidence intervals. Cross-drag induces consistent drops across models and tasks, qualitatively reproducing the self-drag pattern of Figure 2 with somewhat smaller magnitudes.

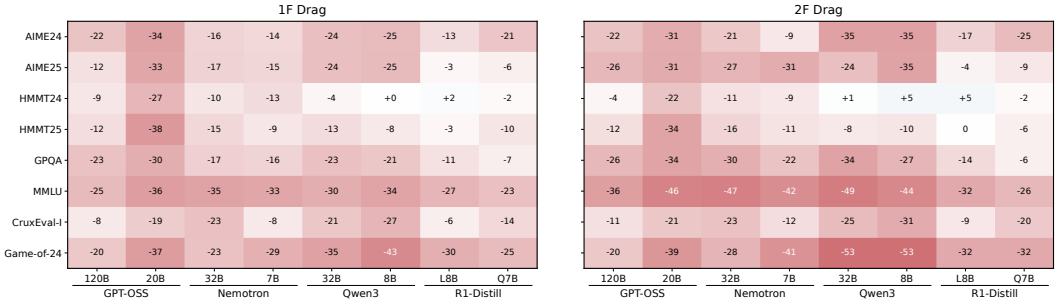

Figure 8: **Cross-drag pass@1 relative to DIRECT.** Heatmap of Δ pass@1 under 1F (left) and 2F (right) when in-context drafts come from other anchor models.

B.3 CROSS-DRAG ON FRONTIER PROPRIETARY MODELS

We additionally evaluate three frontier proprietary models (GPT-5 (OpenAI, 2025b), Gemini-2.5 Pro (Google, 2025a), and Gemini-3 Pro (Google, 2025b)) under the cross-drag setup. Self-drag for these models is impractical: constructing the per-task draft pool and the reasonably-hard subset requires 16 clean-slate samples per problem on each benchmark, which is API-budget-prohibitive in academic settings. The cross-drag setup sidesteps this: we reuse the open-weight anchor pool from Table 9, and only need 5 samples per problem on each proprietary model under DIRECT, 1F, and 2F. We restrict the proprietary evaluation to the four competition math benchmarks (AIME24/25, HMMT24/25), where API costs are still tractable and where the reasonably-hard subset is largest.

Table 12 shows that contextual drag persists into the frontier regime. GPT-5 drops $\geq$ 5pp on AIME25 and HMMT25 under 1F, despite reading drafts from substantially weaker open-weight anchors. Gemini-2.5/3 Pro start from near-saturated DIRECT performance ($\geq$ 94% on three of four tasks, leaving little headroom) and still lose accuracy on AIME24, AIME25, and HMMT24 under 1F or 2F. The magnitudes are smaller than the largest open-weight effects, but the direction is consistent, though magnitudes are small and based on only 5 samples per problem: frontier proprietary models are not immune to incorrect drafts in context.

Table 10: Full per-task pass@1 results for cross-drag evaluation (Section B) with 95% confidence intervals in subscripts. Drafts in context come from other anchor models.

	AIME24*			AIME25*		
	DIRECT	1F	2F	DIRECT	1F	2F
GPT-OSS-120B	66.25 $_{\pm 1.72}$	43.75 $_{\pm 0.51}$ (-22.50)	43.75 $_{\pm 2.04}$ (-22.50)	66.83 $_{\pm 1.47}$	55.29 $_{\pm 1.04}$ (-11.54)	40.38 $_{\pm 1.31}$ (-26.44)
GPT-OSS-20B	51.88 $_{\pm 1.47}$	17.50 $_{\pm 1.60}$ (-34.38)	21.25 $_{\pm 2.32}$ (-30.63)	51.92 $_{\pm 1.55}$	18.75 $_{\pm 0.84}$ (-33.17)	20.67 $_{\pm 1.51}$ (-31.25)
Nemotron-32B	83.75 $_{\pm 1.81}$	67.50 $_{\pm 1.28}$ (-16.25)	63.13 $_{\pm 1.42}$ (-20.62)	78.37 $_{\pm 1.18}$	61.54 $_{\pm 0.91}$ (-16.83)	50.96 $_{\pm 1.47}$ (-27.40)
Nemotron-7B	67.50 $_{\pm 1.28}$	53.12 $_{\pm 1.22}$ (-14.38)	58.75 $_{\pm 1.81}$ (-8.75)	67.79 $_{\pm 1.12}$	52.40 $_{\pm 0.69}$ (-15.38)	37.02 $_{\pm 1.29}$ (-30.77)
Qwen3-32B	65.00 $_{\pm 2.05}$	41.25 $_{\pm 0.84}$ (-23.75)	30.00 $_{\pm 1.55}$ (-35.00)	54.81 $_{\pm 1.47}$	31.25 $_{\pm 0.66}$ (-23.56)	31.25 $_{\pm 0.78}$ (-23.56)
Qwen3-8B	55.62 $_{\pm 2.12}$	30.63 $_{\pm 0.90}$ (-25.00)	20.62 $_{\pm 1.69}$ (-35.00)	45.67 $_{\pm 1.20}$	20.19 $_{\pm 0.77}$ (-25.48)	11.06 $_{\pm 1.04}$ (-34.62)
LlamaR1-8B	21.25 $_{\pm 1.89}$	8.13 $_{\pm 0.94}$ (-13.12)	4.38 $_{\pm 0.77}$ (-16.88)	7.21 $_{\pm 0.86}$	4.33 $_{\pm 0.55}$ (-2.88)	2.88 $_{\pm 0.63}$ (-4.33)
QwenR1-7B	33.13 $_{\pm 2.37}$	11.88 $_{\pm 1.09}$ (-21.25)	8.12 $_{\pm 1.25}$ (-25.00)	9.62 $_{\pm 0.78}$	3.85 $_{\pm 0.37}$ (-5.77)	0.96 $_{\pm 0.35}$ (-8.65)
	HMMT24*			HMMT25*		
	DIRECT	1F	2F	DIRECT	1F	2F
GPT-OSS-120B	48.83 $_{\pm 1.22}$	39.84 $_{\pm 0.60}$ (-8.98)	45.31 $_{\pm 1.07}$ (-3.52)	61.01 $_{\pm 0.70}$	48.51 $_{\pm 0.55}$ (-12.50)	49.11 $_{\pm 0.69}$ (-11.90)
GPT-OSS-20B	40.23 $_{\pm 1.08}$	13.28 $_{\pm 0.83}$ (-26.95)	17.97 $_{\pm 1.21}$ (-22.27)	58.63 $_{\pm 0.78}$	20.54 $_{\pm 0.68}$ (-38.10)	24.70 $_{\pm 1.02}$ (-33.93)
Nemotron-32B	61.72 $_{\pm 1.40}$	51.95 $_{\pm 0.54}$ (-9.77)	50.78 $_{\pm 0.85}$ (-10.94)	80.36 $_{\pm 0.62}$	65.48 $_{\pm 0.55}$ (-14.88)	64.58 $_{\pm 0.70}$ (-15.77)
Nemotron-7B	49.22 $_{\pm 1.14}$	35.94 $_{\pm 0.66}$ (-13.28)	40.62 $_{\pm 0.77}$ (-8.59)	63.10 $_{\pm 0.94}$	54.17 $_{\pm 0.31}$ (-8.93)	52.38 $_{\pm 0.74}$ (-10.71)
Qwen3-32B	39.06 $_{\pm 1.34}$	35.55 $_{\pm 0.43}$ (-3.52)	40.23 $_{\pm 0.76}$ (+1.17)	49.70 $_{\pm 0.82}$	36.90 $_{\pm 0.34}$ (-12.80)	41.37 $_{\pm 0.67}$ (-8.33)
Qwen3-8B	21.88 $_{\pm 0.72}$	22.27 $_{\pm 0.80}$ (+0.39)	26.95 $_{\pm 0.89}$ (+5.08)	42.26 $_{\pm 0.67}$	34.23 $_{\pm 0.46}$ (-8.04)	32.74 $_{\pm 0.67}$ (-9.52)
LlamaR1-8B	5.08 $_{\pm 0.40}$	7.42 $_{\pm 0.38}$ (+2.34)	9.77 $_{\pm 0.66}$ (+4.69)	10.42 $_{\pm 0.68}$	7.44 $_{\pm 0.37}$ (-2.98)	10.42 $_{\pm 0.58}$ (+0.00)
QwenR1-7B	8.59 $_{\pm 0.81}$	6.25 $_{\pm 0.43}$ (-2.34)	6.25 $_{\pm 0.72}$ (-2.34)	19.64 $_{\pm 0.74}$	10.12 $_{\pm 0.38}$ (-9.52)	13.69 $_{\pm 0.69}$ (-5.95)
	GPQA-DIAMOND*			MMLU-REDUX*		
	DIRECT	1F	2F	DIRECT	1F	2F
GPT-OSS-120B	58.24 $_{\pm 0.13}$	35.70 $_{\pm 0.08}$ (-22.54)	31.91 $_{\pm 0.09}$ (-26.33)	67.59 $_{\pm 0.02}$	42.26 $_{\pm 0.02}$ (-25.33)	31.61 $_{\pm 0.03}$ (-35.97)
GPT-OSS-20B	46.97 $_{\pm 0.14}$	16.86 $_{\pm 0.05}$ (-30.11)	13.16 $_{\pm 0.08}$ (-33.81)	58.86 $_{\pm 0.04}$	22.73 $_{\pm 0.01}$ (-36.14)	12.83 $_{\pm 0.02}$ (-46.03)
Nemotron-32B	66.81 $_{\pm 0.08}$	49.43 $_{\pm 0.10}$ (-17.38)	37.22 $_{\pm 0.09}$ (-29.59)	67.25 $_{\pm 0.02}$	32.74 $_{\pm 0.01}$ (-34.50)	20.09 $_{\pm 0.02}$ (-47.16)
Nemotron-7B	51.61 $_{\pm 0.11}$	35.27 $_{\pm 0.07}$ (-16.34)	29.40 $_{\pm 0.07}$ (-22.21)	56.46 $_{\pm 0.02}$	23.58 $_{\pm 0.02}$ (-32.88)	14.02 $_{\pm 0.02}$ (-42.44)
Qwen3-32B	57.62 $_{\pm 0.10}$	34.19 $_{\pm 0.05}$ (-23.44)	23.30 $_{\pm 0.10}$ (-34.33)	70.62 $_{\pm 0.02}$	40.53 $_{\pm 0.02}$ (-30.09)	21.97 $_{\pm 0.02}$ (-48.65)
Qwen3-8B	47.77 $_{\pm 0.09}$	27.13 $_{\pm 0.08}$ (-20.64)	20.45 $_{\pm 0.06}$ (-27.32)	61.65 $_{\pm 0.03}$	27.76 $_{\pm 0.02}$ (-33.90)	17.21 $_{\pm 0.01}$ (-44.44)
LlamaR1-8B	31.53 $_{\pm 0.16}$	20.69 $_{\pm 0.05}$ (-10.84)	17.85 $_{\pm 0.06}$ (-13.68)	41.17 $_{\pm 0.03}$	13.72 $_{\pm 0.02}$ (-27.45)	8.89 $_{\pm 0.02}$ (-32.28)
QwenR1-7B	21.26 $_{\pm 0.08}$	14.73 $_{\pm 0.09}$ (-6.53)	15.20 $_{\pm 0.06}$ (-6.06)	36.60 $_{\pm 0.04}$	13.62 $_{\pm 0.01}$ (-22.98)	10.25 $_{\pm 0.02}$ (-26.35)
	CRUXEVAL-I*			GAME OF 24*		
	DIRECT	1F	2F	DIRECT	1F	2F
GPT-OSS-120B	89.23 $_{\pm 0.09}$	81.02 $_{\pm 0.05}$ (-8.21)	77.89 $_{\pm 0.06}$ (-11.34)	77.34 $_{\pm 0.03}$	56.94 $_{\pm 0.03}$ (-20.40)	57.09 $_{\pm 0.05}$ (-20.24)
GPT-OSS-20B	61.34 $_{\pm 0.30}$	42.39 $_{\pm 0.11}$ (-18.95)	40.53 $_{\pm 0.13}$ (-20.81)	78.30 $_{\pm 0.03}$	41.61 $_{\pm 0.05}$ (-36.70)	39.31 $_{\pm 0.06}$ (-38.99)
Nemotron-32B	77.06 $_{\pm 0.09}$	53.69 $_{\pm 0.06}$ (-23.37)	54.19 $_{\pm 0.09}$ (-22.87)	79.54 $_{\pm 0.03}$	56.68 $_{\pm 0.03}$ (-22.86)	51.53 $_{\pm 0.03}$ (-28.01)
Nemotron-7B	43.55 $_{\pm 0.21}$	35.27 $_{\pm 0.07}$ (-8.28)	31.32 $_{\pm 0.09}$ (-12.23)	75.35 $_{\pm 0.02}$	45.97 $_{\pm 0.03}$ (-29.38)	34.77 $_{\pm 0.02}$ (-40.58)
Qwen3-32B	75.83 $_{\pm 0.08}$	54.89 $_{\pm 0.04}$ (-20.94)	50.90 $_{\pm 0.09}$ (-24.93)	78.48 $_{\pm 0.03}$	43.40 $_{\pm 0.03}$ (-35.09)	25.47 $_{\pm 0.03}$ (-53.01)
Qwen3-8B	67.49 $_{\pm 0.08}$	40.09 $_{\pm 0.06}$ (-27.39)	36.20 $_{\pm 0.08}$ (-31.28)	76.29 $_{\pm 0.03}$	32.92 $_{\pm 0.02}$ (-43.38)	23.26 $_{\pm 0.02}$ (-53.03)
LlamaR1-8B	30.32 $_{\pm 0.08}$	23.84 $_{\pm 0.07}$ (-6.48)	21.81 $_{\pm 0.08}$ (-8.51)	35.34 $_{\pm 0.04}$	5.44 $_{\pm 0.02}$ (-29.90)	3.37 $_{\pm 0.01}$ (-31.97)
QwenR1-7B	36.84 $_{\pm 0.11}$	22.74 $_{\pm 0.06}$ (-14.10)	16.82 $_{\pm 0.06}$ (-20.01)	40.69 $_{\pm 0.03}$	15.60 $_{\pm 0.02}$ (-25.09)	8.49 $_{\pm 0.02}$ (-32.20)

Table 11: Full per-task pass@5 results for cross-drag evaluation (Section B) with 95% confidence intervals in subscripts. Drafts in context come from other anchor models.

	AIME24*			AIME25*		
	DIRECT	1F	2F	DIRECT	1F	2F
GPT-OSS-120B	90.62 $\pm$1.02	68.75 $\pm$0.75 (-21.88)	73.12 $\pm$0.90 (-17.50)	92.31 $\pm$0.74	83.65 $\pm$0.43 (-8.65)	74.52 $\pm$0.88 (-17.79)
GPT-OSS-20B	83.12 $\pm$1.31	40.00 $\pm$0.55 (-43.12)	46.25 $\pm$1.44 (-36.88)	78.85 $\pm$0.69	58.65 $\pm$1.33 (-20.19)	62.98 $\pm$1.44 (-15.87)
Nemotron-32B	100.00 $\pm$0.00	94.38 $\pm$0.60 (-5.63)	92.50 $\pm$1.40 (-7.50)	91.83 $\pm$0.25	88.94 $\pm$0.31 (-2.88)	77.40 $\pm$0.78 (-14.42)
Nemotron-7B	89.38 $\pm$0.38	78.75 $\pm$0.51 (-10.63)	88.12 $\pm$1.13 (-1.25)	87.50 $\pm$0.63	86.54 $\pm$0.52 (-0.96)	63.46 $\pm$0.87 (-24.04)
Qwen3-32B	98.75 $\pm$0.51	67.50 $\pm$0.67 (-31.25)	56.88 $\pm$0.72 (-41.88)	79.33 $\pm$0.48	63.46 $\pm$0.83 (-15.87)	54.33 $\pm$0.45 (-25.00)
Qwen3-8B	76.88 $\pm$1.06	63.75 $\pm$0.51 (-13.12)	54.37 $\pm$1.64 (-22.50)	68.75 $\pm$0.69	51.44 $\pm$0.25 (-17.31)	34.62 $\pm$1.17 (-34.13)
LlamaR1-8B	48.12 $\pm$1.57	26.88 $\pm$0.38 (-21.25)	13.75 $\pm$0.75 (-34.38)	23.08 $\pm$0.64	13.46 $\pm$0.37 (-9.62)	14.42 $\pm$1.38 (-8.65)
QwenR1-7B	59.38 $\pm$0.38	38.75 $\pm$0.64 (-20.62)	31.87 $\pm$2.46 (-27.50)	28.85 $\pm$0.69	18.27 $\pm$0.43 (-10.58)	4.81 $\pm$0.51 (-24.04)
	HMMT24*			HMMT25*		
	DIRECT	1F	2F	DIRECT	1F	2F
GPT-OSS-120B	87.11 $\pm$0.57	68.36 $\pm$0.40 (-18.75)	75.00 $\pm$0.61 (-12.11)	87.20 $\pm$0.50	72.62 $\pm$0.31 (-14.58)	66.67 $\pm$0.48 (-20.54)
GPT-OSS-20B	80.86 $\pm$0.50	40.23 $\pm$0.49 (-40.62)	47.66 $\pm$1.24 (-33.20)	86.01 $\pm$0.85	52.98 $\pm$0.35 (-33.04)	54.76 $\pm$0.48 (-31.25)
Nemotron-32B	89.84 $\pm$0.53	82.81 $\pm$0.33 (-7.03)	85.16 $\pm$0.53 (-4.69)	96.43 $\pm$0.22	86.31 $\pm$0.17 (-10.12)	85.12 $\pm$0.31 (-11.31)
Nemotron-7B	87.11 $\pm$0.63	64.84 $\pm$0.57 (-22.27)	71.48 $\pm$0.71 (-15.62)	91.37 $\pm$0.32	78.57 $\pm$0.25 (-12.80)	78.87 $\pm$0.40 (-12.50)
Qwen3-32B	69.92 $\pm$0.56	63.67 $\pm$0.50 (-6.25)	60.94 $\pm$0.33 (-8.98)	72.92 $\pm$0.47	55.95 $\pm$0.13 (-16.96)	63.69 $\pm$0.40 (-9.23)
Qwen3-8B	51.95 $\pm$0.59	49.22 $\pm$0.57 (-2.73)	51.56 $\pm$0.43 (-0.39)	63.99 $\pm$0.74	51.19 $\pm$0.40 (-12.80)	55.95 $\pm$0.38 (-8.04)
LlamaR1-8B	19.14 $\pm$0.69	21.88 $\pm$0.27 (+2.73)	28.91 $\pm$0.46 (+9.77)	24.70 $\pm$0.63	20.83 $\pm$0.17 (-3.87)	28.27 $\pm$0.55 (+3.57)
QwenR1-7B	27.73 $\pm$0.94	26.56 $\pm$0.51 (-1.17)	25.00 $\pm$0.81 (-2.73)	48.81 $\pm$0.64	29.46 $\pm$0.22 (-19.35)	36.61 $\pm$0.74 (-12.20)
	GPQA-DIAMOND*			MMLU-REDUX*		
	DIRECT	1F	2F	DIRECT	1F	2F
GPT-OSS-120B	84.23 $\pm$0.08	60.27 $\pm$0.05 (-23.96)	53.93 $\pm$0.08 (-30.30)	87.52 $\pm$0.01	62.34 $\pm$0.01 (-25.19)	49.91 $\pm$0.02 (-37.61)
GPT-OSS-20B	80.07 $\pm$0.05	42.90 $\pm$0.08 (-37.17)	32.10 $\pm$0.11 (-47.96)	85.31 $\pm$0.02	47.24 $\pm$0.02 (-38.07)	30.00 $\pm$0.02 (-55.32)
Nemotron-32B	90.34 $\pm$0.04	69.93 $\pm$0.07 (-20.41)	55.26 $\pm$0.09 (-35.09)	87.83 $\pm$0.01	50.74 $\pm$0.01 (-37.09)	32.28 $\pm$0.01 (-55.55)
Nemotron-7B	79.64 $\pm$0.08	59.14 $\pm$0.03 (-20.50)	46.97 $\pm$0.10 (-32.67)	80.72 $\pm$0.02	43.15 $\pm$0.01 (-37.58)	26.93 $\pm$0.01 (-53.79)
Qwen3-32B	79.97 $\pm$0.07	54.31 $\pm$0.04 (-25.66)	37.22 $\pm$0.05 (-42.76)	86.91 $\pm$0.01	61.70 $\pm$0.01 (-25.21)	36.99 $\pm$0.02 (-49.91)
Qwen3-8B	66.76 $\pm$0.09	45.03 $\pm$0.05 (-21.73)	30.40 $\pm$0.05 (-36.36)	77.58 $\pm$0.01	42.18 $\pm$0.01 (-35.40)	29.82 $\pm$0.01 (-47.76)
LlamaR1-8B	64.82 $\pm$0.12	35.94 $\pm$0.07 (-28.88)	27.89 $\pm$0.06 (-36.93)	71.67 $\pm$0.02	30.23 $\pm$0.02 (-41.44)	18.50 $\pm$0.01 (-53.17)
QwenR1-7B	41.71 $\pm$0.08	24.62 $\pm$0.05 (-17.09)	23.82 $\pm$0.08 (-17.90)	64.59 $\pm$0.03	29.54 $\pm$0.02 (-35.05)	21.30 $\pm$0.01 (-43.29)
	CRUXEVAL-I*			GAME OF 24*		
	DIRECT	1F	2F	DIRECT	1F	2F
GPT-OSS-120B	99.14 $\pm$0.01	94.41 $\pm$0.02 (-4.72)	93.28 $\pm$0.03 (-5.85)	97.43 $\pm$0.01	83.65 $\pm$0.02 (-13.77)	84.52 $\pm$0.02 (-12.90)
GPT-OSS-20B	92.15 $\pm$0.03	75.00 $\pm$0.02 (-17.15)	72.31 $\pm$0.05 (-19.85)	98.76 $\pm$0.01	72.78 $\pm$0.03 (-25.98)	73.31 $\pm$0.02 (-25.45)
Nemotron-32B	93.35 $\pm$0.03	78.56 $\pm$0.01 (-14.79)	74.83 $\pm$0.04 (-18.52)	95.65 $\pm$0.01	86.54 $\pm$0.01 (-9.10)	83.43 $\pm$0.01 (-12.21)
Nemotron-7B	80.52 $\pm$0.06	64.49 $\pm$0.06 (-16.02)	60.94 $\pm$0.08 (-19.58)	94.71 $\pm$0.01	80.84 $\pm$0.01 (-13.87)	72.16 $\pm$0.02 (-22.55)
Qwen3-32B	94.05 $\pm$0.02	78.49 $\pm$0.02 (-15.56)	74.27 $\pm$0.04 (-19.78)	97.51 $\pm$0.01	77.56 $\pm$0.02 (-19.96)	51.71 $\pm$0.03 (-45.80)
Qwen3-8B	85.17 $\pm$0.03	61.10 $\pm$0.03 (-24.07)	59.74 $\pm$0.09 (-25.43)	95.33 $\pm$0.01	59.60 $\pm$0.01 (-35.73)	46.41 $\pm$0.03 (-48.92)
LlamaR1-8B	56.85 $\pm$0.06	45.15 $\pm$0.03 (-11.70)	41.32 $\pm$0.06 (-15.53)	69.55 $\pm$0.03	15.97 $\pm$0.01 (-53.58)	9.37 $\pm$0.02 (-60.18)
QwenR1-7B	66.39 $\pm$0.09	43.55 $\pm$0.03 (-22.84)	34.97 $\pm$0.03 (-31.42)	74.44 $\pm$0.03	39.18 $\pm$0.02 (-35.26)	23.43 $\pm$0.02 (-51.01)

Table 12: Cross-drag pass@1 for frontier proprietary models on competition math benchmarks (AIME24/25 and HMMT24/25). Drafts in context come from open-weight anchor models (Table 9). Per-cell numbers are means over 5 samples per problem (fewer than the 16 used for open-weight models, due to API-budget constraints); deltas relative to DIRECT are shown in parentheses. Even on benchmarks where DIRECT accuracy is near saturation, 1F/2F drops persist, indicating that contextual drag is not a quirk of open-weight models.

	AIME24*			AIME25*		
	DIRECT	1F	2F	DIRECT	1F	2F
GPT-5	88.75	88.13 (-0.62)	86.25 (-2.50)	87.50	81.73 (-5.77)	82.69 (-4.81)
Gemini-3 Pro	98.75	98.75 (-0.00)	95.00 (-3.75)	94.23	90.38 (-3.85)	92.31 (-1.92)
Gemini-2.5 Pro	100.00	95.00 (-5.00)	92.50 (-7.50)	94.23	90.38 (-3.85)	92.31 (-1.92)
	HMMT24*			HMMT25*		
	DIRECT	1F	2F	DIRECT	1F	2F
GPT-5	76.56	77.73 (+1.17)	76.56 (-0.00)	91.07	84.82 (-6.25)	85.71 (-5.36)
Gemini-3 Pro	90.62	87.50 (-3.12)	91.41 (+0.79)	99.40	99.40 (-0.00)	98.21 (-1.19)
Gemini-2.5 Pro	90.62	85.94 (-4.68)	92.19 (+1.57)	99.40	100.00 (+0.60)	98.21 (-1.19)

B.4 CROSS-DRAG STRUCTURAL SIMILARITY (TED)

Figure 9 shows that the structural copying behavior observed under self-drag carries over to cross-drag: conditioned generations remain closer in TED to the in-context draft than clean-slate generations are.

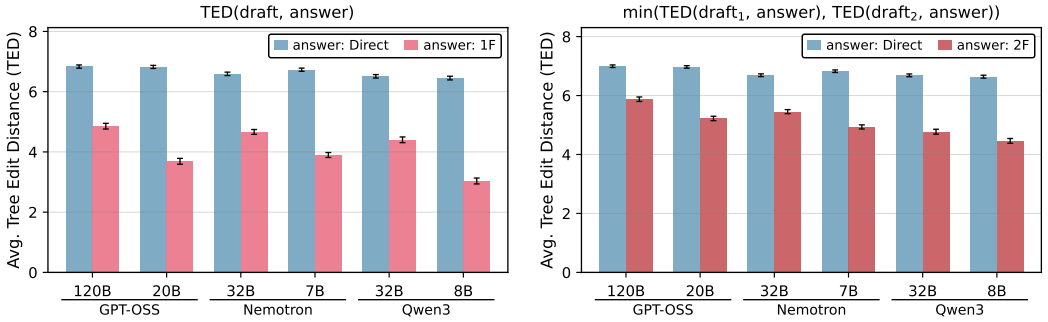

Figure 9: **Cross-drag TED on Game of 24.** Mean tree edit distance between conditioned generations and the in-context draft, compared to clean-slate generations. Lower is more similar to the draft (not better accuracy).

B.5 CROSS-DRAG UNDER ERROR SIGNALS

Figure 10 repeats the error-awareness analysis from Section 3 in the cross-drag setting. As in self-drag, neither external nor self-detected error signals fully recover DIRECT performance.



Figure 10: **Cross-drag pass@1 under error signals.** Heatmaps of 1F pass@1 relative to DIRECT under (left) external and (right) self-detected error signals, computed on cross-drag traces.

B.6 CROSS-DRAG CONTEXT DENOISING

Figure 11 and Tables 13 and 14 report cross-drag analogues of the self-drag context-denoising results in Section F.2. The reader should compare each model's 1F (or 2F) column against its FILTER and REVISE columns within the same task block: the gap between the conditioned column and DIRECT is the cross-drag deficit, and the gap between the denoising columns and DIRECT is the residual deficit after intervention. Tables 13 and 14 additionally include 1F-VER and 2F-VER columns: these are evaluated under the VER (Verification) ablation prompt template (Section C.2), which differs from the standard 1F/2F templates only in that the model is no longer instructed to explicitly verify the in-context draft before producing its solution; they are included here to show that the cross-drag effect persists even when the verification instruction is removed. Qualitatively, both FILTER and REVISE mitigate cross-drag substantially: across most model/task pairs the residual deficit is much smaller than the original drop, often closing more than half of the gap, and in several cases the denoised column meets or slightly exceeds DIRECT. The pattern is broadly consistent

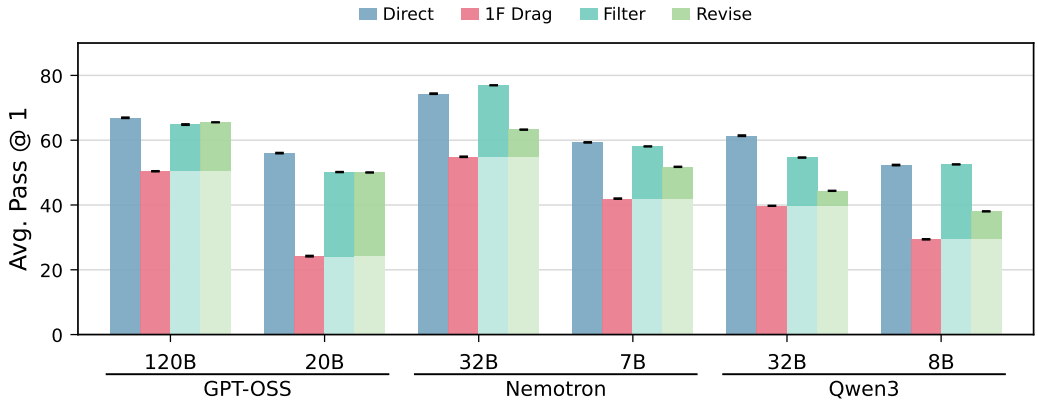

Figure 11: **Test-time context denoising under cross-drag.** For each of the six models used in the cross-drag evaluation, bars show pass@1 (%) under DIRECT, 1F, FILTER, and REVISE, averaged over the eight tasks. Both denoising methods recover most of the cross-drag deficit, and FILTER matches or exceeds DIRECT on Nemotron-32B and Qwen3-8B.

with the self-drag setting, and FILTER is in fact more effective under cross-drag than under self-drag for several models (Figure 11 vs. Figure 6). This supports the interpretation that the recoverable component of contextual drag is driven by the presence of an erroneous trace in context rather than by self-conditioning specifically: denoising is effective regardless of whether the in-context draft was produced by the evaluated model itself or by an anchor model.

Table 13: Additional results of context denoising experiments (1F) with 95% confidence internal.

	AIME24*					AIME25*				
	DIRECT	REVISE	FILTER	1F	1F-VER	DIRECT	REVISE	FILTER	1F	1F-VER
GPT-OSS-120B	66.25 ±1.72	80.00 ±0.55	63.75 ±2.10	43.75 ±0.73	57.50 ±0.95	66.83 ±1.47	68.27 ±1.03	65.38 ±1.52	55.29 ±1.47	58.17 ±1.50
GPT-OSS-20B	51.88 ±1.47	52.50 ±0.77	53.12 ±1.21	17.50 ±2.26	41.25 ±1.80	51.92 ±1.54	48.56 ±1.43	47.60 ±1.22	18.75 ±1.20	46.63 ±2.54
Nemotron-32B	83.75 ±1.80	74.37 ±1.02	84.38 ±1.02	67.50 ±1.82	66.88 ±1.33	78.37 ±1.18	73.56 ±1.07	84.13 ±0.58	61.54 ±1.28	62.50 ±1.21
Nemotron-7B	67.50 ±1.28	63.13 ±1.33	68.12 ±0.94	53.12 ±1.73	56.25 ±2.24	67.79 ±1.12	63.94 ±0.73	70.19 ±0.72	52.40 ±0.97	46.63 ±0.78
Qwen3-32B	65.00 ±2.05	46.25 ±1.31	65.62 ±1.28	41.25 ±1.20	43.75 ±3.04	54.81 ±1.47	37.98 ±0.78	53.85 ±0.98	31.25 ±0.94	34.13 ±1.54
Qwen3-8B	55.62 ±2.12	50.00 ±1.23	66.25 ±1.20	30.63 ±1.28	31.87 ±1.21	45.67 ±1.20	39.42 ±0.96	52.88 ±1.03	20.19 ±1.09	19.23 ±1.23
	HMMT24*					HMMT25*				
	DIRECT	REVISE	FILTER	1F	1F-VER	DIRECT	REVISE	FILTER	1F	1F-VER
GPT-OSS-120B	48.83 ±1.22	57.03 ±0.59	44.92 ±1.18	39.84 ±0.84	43.36 ±1.56	61.01 ±0.70	68.75 ±0.52	59.23 ±0.94	48.51 ±0.78	52.98 ±0.67
GPT-OSS-20B	40.23 ±1.08	50.00 ±0.47	30.86 ±0.42	13.28 ±1.17	30.08 ±0.98	58.63 ±0.78	62.20 ±0.68	45.83 ±0.50	20.54 ±0.97	42.26 ±0.53
Nemotron-32B	61.72 ±1.40	62.11 ±0.99	66.02 ±0.83	51.95 ±0.76	50.78 ±1.00	80.36 ±0.62	74.40 ±0.56	91.96 ±0.47	65.48 ±0.78	67.56 ±0.36
Nemotron-7B	49.22 ±1.15	49.61 ±0.92	43.75 ±0.76	35.94 ±0.94	35.55 ±1.32	63.10 ±0.95	63.10 ±0.47	62.20 ±0.73	54.17 ±0.43	51.19 ±1.30
Qwen3-32B	39.06 ±1.34	41.02 ±0.57	34.77 ±0.78	35.55 ±0.60	29.69 ±0.54	49.70 ±0.83	48.81 ±0.44	43.15 ±0.80	36.90 ±0.47	40.77 ±0.77
Qwen3-8B	21.88 ±0.72	30.47 ±0.52	26.56 ±0.54	22.27 ±1.13	23.05 ±0.85	42.26 ±0.67	38.69 ±0.62	42.86 ±0.51	34.23 ±0.65	36.61 ±0.72
	GPQA-DIAMOND*					MMLU-REDUX*				
	DIRECT	REVISE	FILTER	1F	1F-VER	DIRECT	REVISE	FILTER	1F	1F-VER
GPT-OSS-120B	58.24 ±0.13	46.88 ±0.09	54.88 ±0.14	35.70 ±0.11	38.16 ±0.16	67.59 ±0.02	48.11 ±0.02	63.70 ±0.03	42.26 ±0.03	40.49 ±0.03
GPT-OSS-20B	46.97 ±0.14	33.33 ±0.07	43.70 ±0.16	16.86 ±0.07	26.66 ±0.12	58.86 ±0.04	35.54 ±0.03	55.26 ±0.03	22.73 ±0.01	25.48 ±0.04
Nemotron-32B	66.81 ±0.08	48.15 ±0.06	68.47 ±0.07	49.43 ±0.14	39.44 ±0.08	67.25 ±0.02	31.38 ±0.01	64.54 ±0.01	32.74 ±0.01	24.75 ±0.02
Nemotron-7B	51.61 ±0.11	35.27 ±0.07	55.73 ±0.11	35.27 ±0.10	27.46 ±0.05	56.46 ±0.02	19.63 ±0.01	51.58 ±0.02	23.58 ±0.03	15.99 ±0.02
Qwen3-32B	57.62 ±0.10	27.04 ±0.05	58.38 ±0.06	34.19 ±0.07	22.68 ±0.08	70.62 ±0.02	30.23 ±0.01	39.08 ±0.02	40.53 ±0.03	23.91 ±0.01
Qwen3-8B	47.77 ±0.09	25.24 ±0.05	45.27 ±0.06	27.13 ±0.11	18.75 ±0.06	61.65 ±0.03	28.77 ±0.01	57.96 ±0.02	27.76 ±0.02	20.05 ±0.03
	CRUXEVAL-I*					GAME OF 24*				
	DIRECT	REVISE	FILTER	1F	1F-VER	DIRECT	REVISE	FILTER	1F	1F-VER
GPT-OSS-120B	89.23 ±0.08	81.98 ±0.05	87.80 ±0.06	81.02 ±0.06	87.13 ±0.07	77.34 ±0.03	73.19 ±0.02	78.91 ±0.01	56.94 ±0.05	57.20 ±0.06
GPT-OSS-20B	61.34 ±0.28	61.24 ±0.10	49.73 ±0.14	42.39 ±0.14	45.11 ±0.12	78.30 ±0.03	56.93 ±0.05	75.20 ±0.08	41.61 ±0.07	39.76 ±0.07
Nemotron-32B	77.06 ±0.09	67.49 ±0.06	73.80 ±0.06	53.69 ±0.08	52.59 ±0.13	79.54 ±0.03	74.62 ±0.01	82.30 ±0.01	56.68 ±0.04	51.00 ±0.02
Nemotron-7B	43.55 ±0.20	48.34 ±0.06	39.13 ±0.08	35.27 ±0.09	35.34 ±0.09	75.35 ±0.02	71.18 ±0.02	73.96 ±0.02	45.97 ±0.04	38.97 ±0.03
Qwen3-32B	75.83 ±0.08	65.36 ±0.03	62.90 ±0.06	54.89 ±0.06	53.96 ±0.04	78.48 ±0.03	58.26 ±0.02	79.39 ±0.03	43.40 ±0.05	41.62 ±0.03
Qwen3-8B	67.49 ±0.07	49.00 ±0.06	51.80 ±0.09	40.09 ±0.08	38.33 ±0.04	76.29 ±0.03	42.71 ±0.02	76.67 ±0.02	32.92 ±0.03	29.34 ±0.02

Table 14: Additional results of context denoising experiments (2F) with 95% confidence internal.

	AIME24*					AIME25*				
	DIRECT	REVISE	FILTER	2F	2F-VER	DIRECT	REVISE	FILTER	2F	2F-VER
GPT-OSS-120B	66.25 ±1.72	61.87 ±1.25	57.50 ±1.60	43.75 ±2.04	55.00 ±2.19	66.83 ±1.47	68.27 ±0.73	54.33 ±1.45	40.38 ±1.30	56.73 ±1.52
GPT-OSS-20B	51.88 ±1.47	47.50 ±1.16	49.38 ±2.01	21.25 ±2.32	40.62 ±2.01	51.92 ±1.54	54.33 ±1.01	50.48 ±1.22	20.67 ±1.51	46.15 ±1.11
Nemotron-32B	83.75 ±1.80	70.62 ±1.02	83.12 ±1.31	63.13 ±1.42	60.62 ±1.39	78.37 ±1.18	71.63 ±0.80	71.63 ±0.71	50.96 ±1.47	63.46 ±1.26
Nemotron-7B	67.50 ±1.28	64.38 ±0.77	71.25 ±1.21	58.75 ±1.80	51.88 ±1.25	67.79 ±1.12	48.56 ±0.71	75.96 ±0.97	37.02 ±1.29	41.83 ±1.28
Qwen3-32B	65.00 ±2.05	35.62 ±1.34	60.62 ±1.02	30.00 ±1.55	47.50 ±1.69	54.81 ±1.47	38.46 ±0.98	58.65 ±0.73	31.25 ±0.78	30.29 ±1.01
Qwen3-8B	55.62 ±2.12	34.38 ±0.94	65.62 ±1.22	20.62 ±1.69	24.38 ±1.22	45.67 ±1.20	15.38 ±0.64	43.27 ±0.97	11.06 ±1.04	15.38 ±1.33
	HMMT24*					HMMT25*				
	DIRECT	REVISE	FILTER	2F	2F-VER	DIRECT	REVISE	FILTER	2F	2F-VER
GPT-OSS-120B	48.83 ±1.22	59.38 ±0.38	48.05 ±1.14	45.31 ±1.06	45.31 ±0.99	61.01 ±0.70	59.82 ±0.51	57.44 ±0.66	49.11 ±0.69	55.36 ±0.82
GPT-OSS-20B	40.23 ±1.08	57.03 ±1.11	50.39 ±1.10	17.97 ±1.21	35.94 ±0.88	58.63 ±0.78	56.85 ±0.66	52.68 ±0.77	24.70 ±1.02	44.05 ±0.89
Nemotron-32B	61.72 ±1.40	59.77 ±0.47	67.97 ±1.01	50.78 ±0.85	47.66 ±0.89	80.36 ±0.62	76.19 ±0.44	79.46 ±0.50	64.58 ±0.70	62.50 ±0.69
Nemotron-7B	49.22 ±1.15	56.64 ±0.99	57.42 ±0.87	40.62 ±0.76	32.42 ±0.95	63.10 ±0.95	68.45 ±0.30	75.30 ±0.45	52.38 ±0.74	49.11 ±0.78
Qwen3-32B	39.06 ±1.34	29.30 ±0.75	33.20 ±0.80	40.23 ±0.76	26.17 ±0.82	49.70 ±0.83	47.02 ±0.50	50.60 ±0.62	41.37 ±0.67	34.23 ±0.81
Qwen3-8B	21.88 ±0.72	26.95 ±0.45	25.78 ±0.46	26.95 ±0.89	18.36 ±0.96	42.26 ±0.67	40.48 ±0.44	44.35 ±0.78	32.74 ±0.67	32.74 ±0.72
	GPQA-DIAMOND*					MMLU-REDUX*				
	DIRECT	REVISE	FILTER	2F	2F-VER	DIRECT	REVISE	FILTER	2F	2F-VER
GPT-OSS-120B	58.24 ±0.13	43.56 ±0.11	53.17 ±0.15	31.91 ±0.09	31.20 ±0.08	67.59 ±0.02	46.67 ±0.02	63.43 ±0.02	31.61 ±0.03	30.12 ±0.02
GPT-OSS-20B	46.97 ±0.14	30.87 ±0.12	42.28 ±0.18	13.16 ±0.08	20.27 ±0.14	58.86 ±0.04	35.19 ±0.01	53.71 ±0.02	12.83 ±0.02	14.95 ±0.03
Nemotron-32B	66.81 ±0.08	46.64 ±0.05	63.87 ±0.05	37.22 ±0.09	34.04 ±0.09	67.25 ±0.02	30.71 ±0.01	67.43 ±0.01	20.09 ±0.02	17.03 ±0.02
Nemotron-7B	51.61 ±0.11	35.80 ±0.05	49.72 ±0.08	29.40 ±0.07	27.56 ±0.06	56.46 ±0.02	21.10 ±0.01	51.56 ±0.01	14.02 ±0.02	11.63 ±0.02
Qwen3-32B	57.62 ±0.10	26.09 ±0.03	31.72 ±0.07	23.30 ±0.10	20.08 ±0.08	70.62 ±0.02	26.58 ±0.01	66.93 ±0.02	21.97 ±0.02	14.69 ±0.01
Qwen3-8B	47.77 ±0.09	26.42 ±0.03	46.59 ±0.06	20.45 ±0.06	17.99 ±0.06	61.65 ±0.03	28.15 ±0.01	57.71 ±0.02	17.21 ±0.01	12.63 ±0.01
	CRUXEVAL-I*					GAME OF 24*				
	DIRECT	REVISE	FILTER	2F	2F-VER	DIRECT	REVISE	FILTER	2F	2F-VER
GPT-OSS-120B	89.23 ±0.08	75.96 ±0.09	79.06 ±0.12	77.89 ±0.06	83.54 ±0.09	77.34 ±0.03	72.27 ±0.02	78.81 ±0.03	57.09 ±0.05	57.78 ±0.04
GPT-OSS-20B	61.34 ±0.28	61.90 ±0.13	44.68 ±0.18	40.53 ±0.12	44.18 ±0.12	78.30 ±0.03	56.07 ±0.04	76.37 ±0.05	39.31 ±0.06	40.02 ±0.07
Nemotron-32B	77.06 ±0.09	72.61 ±0.07	71.34 ±0.06	54.19 ±0.09	51.80 ±0.06	79.54 ±0.03	74.41 ±0.02	81.35 ±0.02	51.53 ±0.03	50.63 ±0.03
Nemotron-7B	43.55 ±0.20	47.91 ±0.07	38.36 ±0.06	31.32 ±0.09	36.10 ±0.12	75.35 ±0.02	70.55 ±0.01	73.62 ±0.02	34.77 ±0.02	39.11 ±0.04
Qwen3-32B	75.83 ±0.08	69.32 ±0.04	62.23 ±0.07	50.90 ±0.08	51.66 ±0.09	78.48 ±0.03	56.99 ±0.02	78.18 ±0.03	25.47 ±0.03	41.54 ±0.03
Qwen3-8B	67.49 ±0.07	53.86 ±0.04	49.80 ±0.06	36.20 ±0.08	36.20 ±0.06	76.29 ±0.03	43.78 ±0.01	76.21 ±0.01	23.26 ±0.02	29.24 ±0.03

C Ablation Evaluation of Contextual Drag

C.1 Thinking Budget

This subsection expands on the thinking-budget ablation introduced in the ablation paragraph of Section 2.1. We further investigate whether contextual drag is sensitive to the model's thinking budget. We evaluate GPT-OSS-20B and GPT-OSS-120B at the default reasoning effort and at a HIGH-131K configuration (high reasoning effort with a 131k-token thinking budget). We re-evaluate the DIRECT, 1F, and 2F settings on four core benchmarks (AIME25*, HMMT25*, GPQA-Diamond*, CRUXEval-I*) at both budget levels.

Table 15: **Effect of thinking budget on contextual drag.** Pass@1 (mean ± std across trajectories, in pp) for GPT-OSS-20B and GPT-OSS-120B at the default reasoning effort and at the HIGH-131K configuration (high reasoning effort with a 131k-token thinking budget). For each task we report DIRECT alongside the two anchored variants 1F and 2F.

Model	AIME25*			HMMT25*		
	DIRECT	1F	2F	DIRECT	1F	2F
GPT-OSS-20B	51.92 ±2.84	18.75 ±2.20	20.67 ±2.78	58.63 ±1.82	20.54 ±2.26	24.70 ±2.39
GPT-OSS-120B	66.83 ±2.70	55.29 ±2.71	40.38 ±2.40	61.01 ±1.64	48.51 ±1.83	49.11 ±1.61
GPT-OSS-20B (High-131k)	83.65 ±1.63	75.96 ±2.10	73.08 ±4.30	86.31 ±2.30	67.86 ±1.79	79.76 ±1.83
GPT-OSS-120B (High-131k)	94.71 ±1.31	90.38 ±2.04	87.50 ±1.65	93.45 ±1.18	89.29 ±1.33	81.55 ±1.62
Model	GPQA-DIAMOND*			CRUXEVAL-I*		
	DIRECT	1F	2F	DIRECT	1F	2F
GPT-OSS-20B	46.97 ±0.80	16.86 ±0.42	13.16 ±0.48	69.12 ±1.63	51.86 ±1.74	54.19 ±1.22
GPT-OSS-120B	58.24 ±0.74	35.70 ±0.62	31.91 ±0.55	95.58 ±0.25	85.84 ±0.35	83.91 ±0.41
GPT-OSS-20B (High-131k)	56.34 ±0.95	30.49 ±0.68	26.70 ±1.34	92.59 ±0.48	79.65 ±1.00	83.59 ±0.72
GPT-OSS-120B (High-131k)	68.99 ±0.42	47.63 ±0.72	43.18 ±0.80	96.86 ±0.30	86.57 ±0.47	89.57 ±0.54

As shown in Table 15, increasing the thinking budget substantially improves DIRECT accuracy on most tasks (most prominently on AIME25* and HMMT25*) and shrinks the contextual-drag gap on these tasks. Nevertheless, drag is not eliminated: both 1F and 2F continue to trail DIRECT in nearly every cell, and on GPQA-Diamond* the gap remains comparable at both budget levels. This indicates that while a larger thinking budget can mitigate contextual drag, it does not on its own remove the effect.

C.2 Position and Verification

We provide the full details of implementation of the **Position** (POS) and **Verification** (VER) ablations below. To enable a fair comparison with the 1F results in Figure 2, all ablation prompt templates are adapted from 1F template with minimal changes. For easy reference, we highlight those changes in blue color.

Position This ablation investigates the effect of varying the position of the question and the draft solution. Liu et al. (2024) points out that the model tends to make more use of the information at the beginning or end of the input context, which is known as the "lost-in-the-middle" effect. We adjust the position of (1) the question to the beginning and (2) the draft to the middle of the prompt. We expect that this change in position can reduce the use of the in-context draft, therefore alleviating contextual drag.

POS *ablation (*1F*)*

```
In this task, you will be provided with a problem and one draft
solution to that problem.  The problem is as follows:
-- beginning of problem --
{problem}
-- end of problem --
```

```
For the draft, walk through the solution step-by-step and check for
any fatal mistakes in the reasoning (for example:  incorrect use of
theorems or facts, incorrect calculations, etc.).  Please note that
once you spot a fatal error in a draft's reasoning, you **MUST** stop
checking the remaining steps of that draft and move on.

For the draft, output an overall verdict on the correctness of the
draft using the following tag format:

<overall_verdict>[Correct / Incorrect]</overall_verdict>

While evaluating individual steps, be strict about algebra, logic, and
correct theorem application.  Here is the draft solution you need to
consider:

### Solution:
-- beginning of the draft --
{draft1}
-- end of the draft --

If the draft is incorrect, write a complete, correct solution (you may
reuse sound ideas from the draft).  If the draft is correct, you may
adopt and polish it as the final solution.

Conclude by presenting the final answer only inside \boxed{} (i.e., the
final answer should appear only inside a LaTeX '\boxed{}').

Please:
1.  Carefully evaluate the draft solution step-by-step.  Once you spot
a fatal error in a draft's reasoning, you **MUST** stop checking the
remaining steps of that draft and move on.
2.  Output the correctness verdicts for the draft.
3.  Provide a correct solution to the problem with complete reasoning
steps that lead to the answer.

Remember:  conclude with the final answer only in \boxed{}.
```

Verification This ablation investigates the effect of performing an explicit check of the draft before generating the solution. We remove the corresponding instructions in the input prompt so that the model does not need to perform the explicit verification. While an explicit verification can help the model become aware of the mistakes contained in the draft, this also results in more verbose responses and therefore longer contexts that can lead to performance degradation.

VER *ablation (single draft)*

```
In this task, you will be provided with a problem and one draft
solution to that problem.  If the draft is incorrect, write a complete,
correct solution (you may reuse sound ideas from the draft).  If the
draft is correct, you may adopt and polish it as the final solution.

Conclude by presenting the final answer only inside \boxed{} (i.e., the
final answer should appear only inside a LaTeX '\boxed{}').

The problem is as follows:
-- beginning of problem --
{problem}
-- end of problem --

Here is the draft solution you need to consider:

### Solution:
-- beginning of the draft --
{draft1}
```

```
-- end of the draft --

Please provide a correct solution to the problem with complete
reasoning steps that lead to the answer.

Remember:  conclude with the final answer only in \boxed{}.
```

Table 16 shows the results of the **Position** (POS) and **Verification** (VER) ablation evaluation of GPT-OSS-120B and Nemotron-7B/32B, which exhibit more robustness to contextual drag in Figure 2, on HMMT25, GPQA, and CRUXEval-I.

Table 16: **Position and Verification ablations**: DIRECT and 1F statistics are copied from Figure 2 for comparison. POS and VER denote to the **Position** and **Verification** ablations.

	HMMT25*			
	DIRECT	POS	VER	1F
GPT-OSS-120B	61.01±0.70	53.57±0.77	52.98±0.67	48.51±0.78
Nemotron-32B	80.36±0.62	73.81±0.88	67.56±0.36	65.48±0.78
Nemotron-7B	63.10±0.95	52.38±0.36	51.19±1.30	54.17±0.43
	GPQA-DIAMOND*			
	DIRECT	POS	VER	1F
GPT-OSS-120B	58.24±0.13	42.52±0.09	38.16±0.16	35.70±0.11
Nemotron-32B	66.81±0.08	49.95±0.08	39.44±0.08	49.43±0.14
Nemotron-7B	51.61±0.11	37.59±0.12	27.46±0.05	35.27±0.10
	CRUXEVAL-I*			
	DIRECT	POS	VER	1F
GPT-OSS-120B	89.23±0.08	55.49±0.14	87.13±0.07	81.02±0.06
Nemotron-32B	77.06±0.09	52.76±0.09	52.59±0.13	53.69±0.08
Nemotron-7B	43.55±0.20	32.81±0.13	35.34±0.09	35.27±0.09

Across all three benchmarks, neither POS nor VER fully recovers the DIRECT performance: a gap between 1F and DIRECT persists under both ablations. On HMMT25 and GPQA-Diamond, POS reduces drag more effectively than VER; on CRUXEval-I, the two ablations behave more similarly and can fall below 1F for some models.

C.3 MATCHED-PROMPT CONTROL

A possible concern is that the 1F/2F prompt, which asks the model to verify each draft and output a verdict before re-solving, is itself out of distribution for reasoning models and degrades performance regardless of the draft. We therefore run a matched-prompt control that reuses the DIRECT prompt verbatim and appends the incorrect draft(s) after the problem, with no verification or reuse instruction (matched-prompt template). Subsets, drafts, and DIRECT numbers are identical to the main self-drag evaluation (Table 3).

Matched prompt (one draft)

```
{problem} Let's think step by step and please wrap your final answer in
\boxed{}.

### Draft Solution:
-- beginning of the draft --
{draft1}
-- end of the draft --
```

Tables 17 and 18 show that the degradation persists under the matched prompt. In self-drag, appending incorrect drafts lowers pass@1 in 35 of 36 (model, task) cells for both one and two drafts, by 26.3 and

26.1 points on average, compared with 23.4 and 26.9 points under the 1F/2F prompt on the same cells. The drops are largest on GPQA-DIAMOND* (−36.0 with one draft) and CRUXEVAL-I* (−30.4). The only exceptions are GPT-OSS-120B on CRUXEVAL-I* with one draft (+1.2) and Gemma4-31B on AIME25* with two drafts (+8.8). In cross-drag, all 48 cells degrade, by 29.9 (one draft) and 29.6 points (two drafts) on average. Appending an incorrect draft degrades performance even without any verification instruction, so the effect does not require the verify-then-solve interface. The interface modulates magnitude in both directions: it amplifies drag for GPT-OSS-20B on competition math (AIME25*: −42.9 under 1F vs. −12.9 matched; HMMT25*: −37.9 vs. −13.6) and attenuates it on GPQA-DIAMOND* and CRUXEVAL-I*. The 1F-style prompt also improves performance when the in-context draft is correct (Section C.4); this argues against the verification-style format being harmful in general, but does not show that the effect is specific to the draft's content.

Table 17: **Matched-prompt control, self-drag** (pass@1, %). The DIRECT prompt is reused verbatim with the model's own incorrect draft(s) appended and no verification instruction (matched-prompt template); subsets and DIRECT numbers are identical to Table 3. Parentheses give the change relative to DIRECT. The last row repeats the mean change under the original 1F/2F prompt on the same models and tasks. Gemma4-E4B is not included in this control.

	AIME25*			HMMT25*			GPQA-DIAMOND*			CRUXEVAL-I*		
	DIRECT	+1 draft	+2 drafts	DIRECT	+1 draft	+2 drafts	DIRECT	+1 draft	+2 drafts	DIRECT	+1 draft	+2 drafts
GPT-OSS-120B	63.19	58.82 (-4.37)	59.03 (-4.16)	58.09	33.82 (-24.27)	42.28 (-15.81)	57.47	28.87 (-28.60)	26.77 (-30.70)	81.40	82.59 (+1.19)	80.40 (-1.00)
GPT-OSS-20B	66.96	54.02 (-12.94)	62.95 (-4.01)	55.51	41.91 (-13.60)	33.09 (-22.42)	55.03	37.21 (-17.82)	33.65 (-21.38)	70.37	58.49 (-11.88)	57.91 (-12.46)
Nemotron-32B	56.25	45.83 (-10.42)	54.17 (-2.08)	66.96	47.32 (-19.64)	54.46 (-12.50)	55.94	12.19 (-43.75)	13.05 (-42.89)	72.55	35.36 (-37.19)	32.84 (-39.71)
Nemotron-7B	58.04	39.29 (-18.75)	40.18 (-17.86)	52.23	33.93 (-18.30)	42.86 (-9.37)	50.06	12.74 (-37.32)	11.73 (-38.33)	55.15	13.25 (-41.90)	9.98 (-45.17)
Qwen3-32B	70.45	26.14 (-44.31)	39.20 (-31.25)	48.86	18.75 (-30.11)	18.18 (-30.68)	50.26	4.77 (-45.49)	6.42 (-43.84)	66.49	28.81 (-37.68)	27.49 (-39.00)
Qwen3-8B	68.75	37.50 (-31.25)	28.65 (-40.10)	46.88	27.34 (-19.54)	23.44 (-23.44)	56.08	6.68 (-49.40)	7.02 (-49.06)	62.84	21.10 (-41.74)	18.98 (-43.86)
LlamaR1-8B	50.00	18.06 (-31.94)	31.25 (-18.75)	48.96	30.21 (-18.75)	19.79 (-29.17)	46.85	13.69 (-33.16)	13.78 (-33.07)	52.35	21.96 (-30.39)	23.73 (-28.62)
QwenR1-7B	44.23	28.85 (-15.38)	24.04 (-20.19)	32.81	9.90 (-22.91)	14.06 (-18.75)	43.83	9.62 (-34.21)	7.65 (-36.18)	57.28	19.24 (-38.04)	20.86 (-36.42)
Gemma4-31B	66.25	65.00 (-1.25)	75.00 (+8.75)	69.64	58.93 (-10.71)	40.18 (-29.46)	54.88	20.58 (-34.30)	17.84 (-37.04)	73.21	37.50 (-35.71)	34.38 (-38.83)
Mean Δ (matched prompt)		-18.96	-14.41		-19.76	-21.29		-36.01	-36.94		-30.37	-31.67
Mean Δ (1F/2F prompt)		-22.49	-25.57		-21.78	-25.12		-28.55	-33.94		-20.60	-22.86

Table 18: **Matched-prompt control, cross-drag** (pass@1, %). Same prompt as Table 17, with incorrect drafts drawn from the anchor models of Section B. Parentheses give the change relative to DIRECT.

	AIME25*			HMMT25*			GPQA-DIAMOND*			CRUXEVAL-I*		
	DIRECT	+1 draft	+2 drafts	DIRECT	+1 draft	+2 drafts	DIRECT	+1 draft	+2 drafts	DIRECT	+1 draft	+2 drafts
GPT-OSS-120B	79.09	54.33 (-24.76)	53.85 (-25.24)	62.78	52.38 (-10.40)	47.62 (-15.16)	68.56	33.62 (-34.94)	29.64 (-38.92)	96.86	84.18 (-12.68)	84.06 (-12.80)
GPT-OSS-20B	67.31	49.04 (-18.27)	50.00 (-17.31)	60.51	50.00 (-10.51)	55.36 (-5.15)	57.58	28.17 (-29.41)	22.73 (-34.85)	92.59	49.77 (-42.82)	48.82 (-43.77)
Nemotron-32B	85.10	64.42 (-20.68)	58.65 (-26.45)	81.25	66.67 (-14.58)	62.50 (-18.75)	74.55	37.45 (-37.10)	34.28 (-40.27)	93.27	53.59 (-39.68)	58.35 (-34.92)
Nemotron-7B	79.81	53.85 (-25.96)	45.19 (-34.62)	64.77	47.62 (-17.15)	51.79 (-12.98)	61.66	31.16 (-30.50)	29.92 (-31.74)	55.13	34.44 (-20.69)	39.57 (-15.56)
Qwen3-32B	71.15	25.48 (-45.67)	33.65 (-37.50)	51.99	35.12 (-16.87)	44.64 (-7.35)	68.36	18.66 (-49.70)	19.70 (-48.66)	94.11	47.07 (-47.04)	46.92 (-47.19)
Qwen3-8B	65.87	11.54 (-54.33)	13.46 (-52.41)	44.89	27.98 (-16.91)	39.29 (-5.60)	59.62	17.80 (-41.82)	17.90 (-41.72)	90.91	35.31 (-55.60)	29.62 (-61.29)
Mean Δ (matched prompt)		-31.61	-32.26		-14.40	-10.83		-37.24	-39.36		-36.42	-35.92

C.4 SCALING: VARYING THE NUMBERS OF IN-CONTEXT DRAFTS WITH MIXED CORRECTNESS

This subsection expands on the mixed-correctness ablation introduced in the ablation paragraph of Section 2.1. We ablate the number of drafts in context and their correctness and observe the corresponding effects on contextual drag. Specifically, we repeat the same evaluation in Section 2.1 with 1-4 in-context drafts with mixed correctness (1T, 1F; 2T, 1T1F, 2F; 3T, 2T1F, 1T2F, 3F; and 4T, 3T1F, 2T2F, 1T3F, 4F).

As shown in Figure 12, performance degrades sharply when the context contains only incorrect drafts, with more severe drops as the number of incorrect drafts increases. Correct drafts in context generally result in large performance gains, but models can still struggle when incorrect drafts dominate the context (Figure 12), notably for GPT-OSS-120B and Nemotron-32B on CRUXEval-I and GPQA.

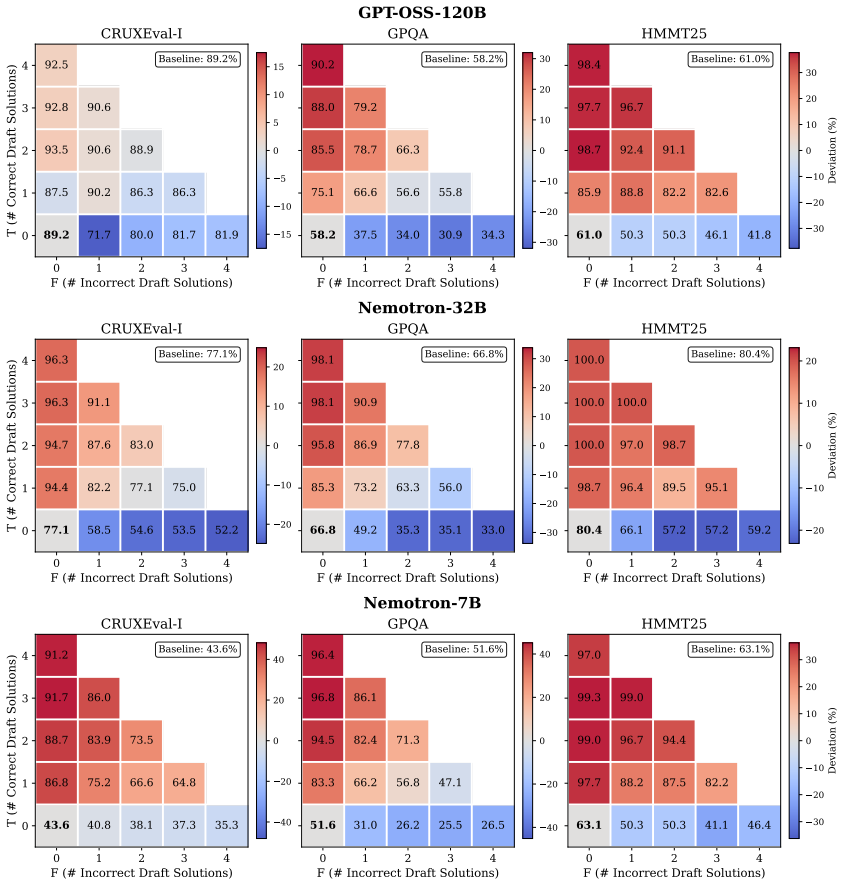

Figure 12: **Scaling ablation**: Performance changes relative to the DIRECT performance as the number of drafts with mixed correctness in context increases.

D TREE-EDIT DISTANCE (TED) ANALYSIS

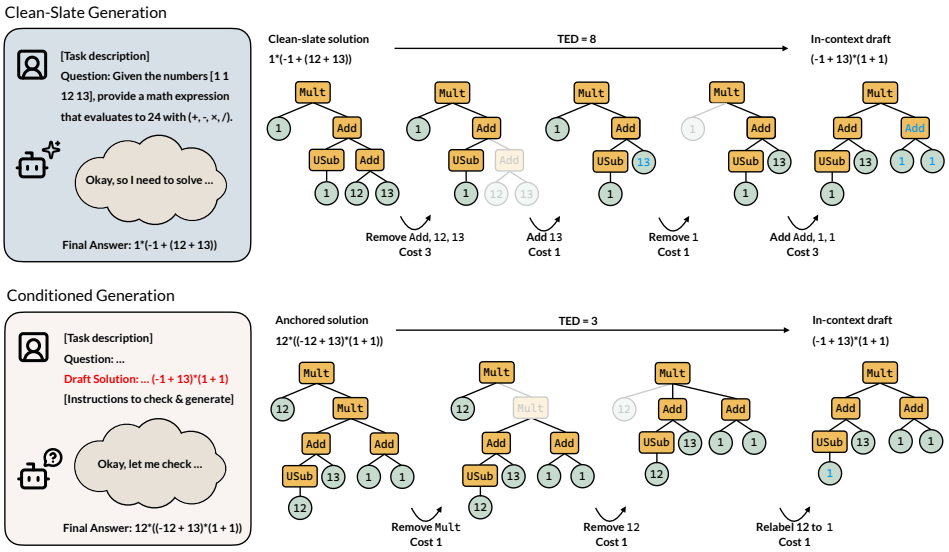

Figure 13: **Overview of TED analysis**: We measure the similarity in reasoning structure of the clean-slate generation, in-context draft, and the conditioned generation using TED. For each pair of clean-slate/conditioned generation and in-context draft, we extract the solution and parse it into a tree, then compute the TED metric between each pair.

D.1 IMPLEMENTATION

We conduct tree edit distance (TED) analysis on the synthetic Game of 24 task, which is commonly referred to as a restricted variant of the *Countdown* problem. Since each solution to the Game of 24 puzzle can be parsed into a tree, this allows us to quantify the structural similarity of the reasoning and error patterns underlying the solution.

We take the set of clean-slate generations under DIRECT and the conditioned generations under 1F in Section 2.1. As shown in Figure 13, for each problem, we form two pairs of responses: (1) a clean-slate generation and an in-context draft solution, and (2) a conditioned generation and an in-context draft solution. For each pair, we extract the solutions from the reasoning traces and parse them into trees. Then we compute the TED between each pair.

In the original formulation, the TED metric measures the dissimilarity between two trees by the minimum-cost sequence of edit operations needed to transform one tree into the other Zhang & Shasha (1989). The computation of TED involves three types of operations: (1) node insertion, (2) node deletion, and (3) node relabeling (or substitution). Figure 13 provides an example of TED computation used in our analysis.

D.2 DEFINITION OF 2F TED

We also analyze the conditioned generations under 2F, whose results are reported alongside 1F in Figure 4 (right panel). Since there are two in-context drafts, we define

$$\text{TED}'(\text{generation}, \{\text{draft1}, \text{draft2}\}) = \min(\text{TED}(\text{generation}, \text{draft1}), \text{TED}(\text{generation}, \text{draft2})).$$

This formulation necessarily yields lower absolute TED values, but does not imply that contextual drag is more severe under 2F than under 1F. Since we apply this definition to both clean-slate and conditioned generations, the comparison between their TED values remains fair.

D.3 Ablation of Answer Reuse Instruction

Another potential confound for the reduced TED distance is the explicit instruction that prompts the model to reuse parts of the solution if it finds them correct. In this ablation setting, we show that removing such reuse instruction does not eliminate the edit distance gap. Below we highlight the removal of ~~the reuse instruction in the prompt~~.

1F *(no reuse instruction)*

```
In this task, you will be provided with a problem and one draft
solution to that problem.  For the draft, walk through the solution
step-by-step and check for any fatal mistakes in the reasoning (for
example:  incorrect use of theorems or facts, incorrect calculations,
etc.).  Please note that once you spot a fatal error in a draft's
reasoning, you **MUST** stop checking the remaining steps of that draft
and move on.

For the draft, output an overall verdict on the correctness of the
draft using the following tag format:

<overall_verdict>[Correct / Incorrect]</overall_verdict>

While evaluating individual steps, be strict about algebra, logic,
and correct theorem application.  If the draft is incorrect, write a
complete, correct solution~~ (you may reuse sound ideas from the draft)~~.
If the draft is correct, you may adopt and polish it as the final
solution.

Conclude by presenting the final answer only inside \boxed{} (i.e., the
final answer should appear only inside a LaTeX `\boxed{}`).

The problem is as follows:
-- beginning of problem --
{problem}
-- end of problem --

Here is the draft solution you need to consider:

### Solution:
-- beginning of the draft --
{draft1}
-- end of the draft --

Please:
1.  Carefully evaluate the draft solution step-by-step.  Once you spot
a fatal error in a draft's reasoning, you **MUST** stop checking the
remaining steps of that draft and move on.
2.  Output the correctness verdicts for the draft.
3.  Provide a correct solution to the problem with complete reasoning
steps that lead to the answer.

Remember:  conclude with the final answer only in \boxed{}.
```

As shown in Figure 14, removing the reuse instruction does not significantly change the trend, which remains consistent with Figure 4 and Figure 5. The left panel of Figure 14 reports TED across all generations, while the right panel restricts the comparison to trajectories where the model correctly identifies the draft as incorrect; the TED gap between conditioned and clean-slate generations persists in both. This further supports our interpretation that the reduced TED distance is driven by contextual drag rather than instruction-following behavior.

D.4 TED Analysis Under Error Signals

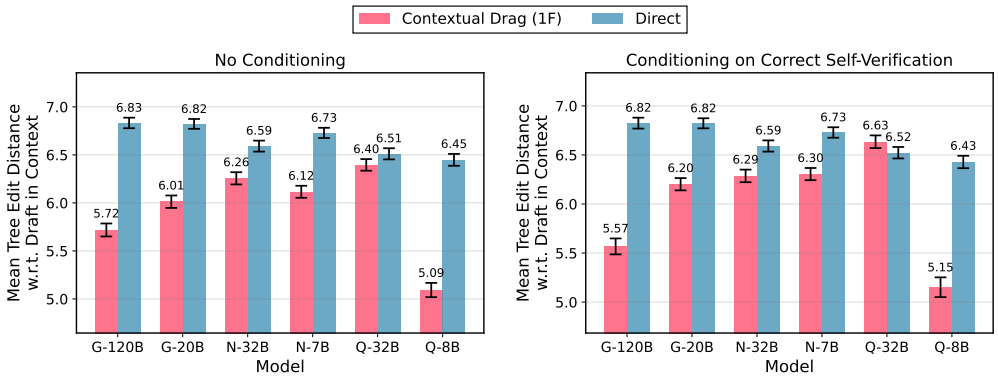

Figure 14: **TED analysis of conditioned generations under 1F without reuse instruction in prompt**

Under the external-error-signal prompt of Section 3.1, the TED comparison still shows that conditioned generations remain structurally closer to the draft than clean-slate generations are (Figure 15, left). This is consistent with the persistent pass@1 drop in Figure 5 (left). The right panel of Figure 15 reports the analogous comparison restricted to trajectories where the model itself emits a correct "Incorrect" verdict (Section 3.2); the TED gap persists in this subset as well.

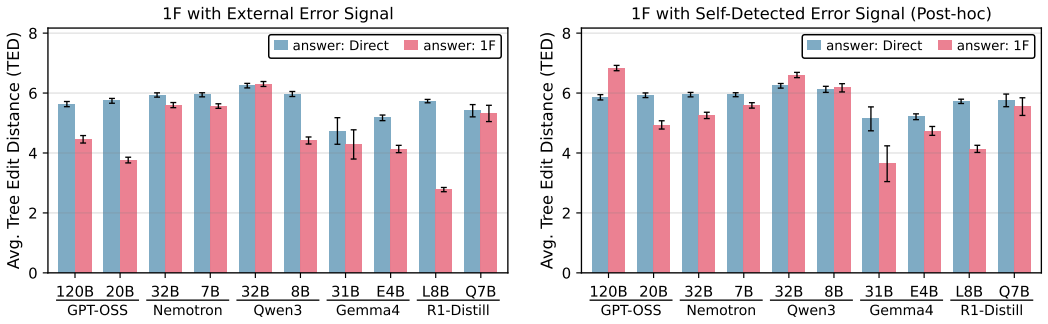

Figure 15: **TED on Game of 24 under error signals.** Mean tree edit distance between conditioned generations and the in-context draft, compared to clean-slate generations. Left: external error signal, where the prompt explicitly labels the draft as incorrect. Right: self-detected error signal (post-hoc), restricting to trajectories where the model itself emits a correct "Incorrect" verdict. Lower is more similar to the draft (not better accuracy).

E Additional Analysis of Contextual Drag Despite Error Awareness

External Error Signal. Here we provide the modified prompt used in Section 3.1 which explicitly and repeatedly stated the incorrectness of the draft in context.

External Error Signal (1F)

```
In this task, you will be provided with a problem and one draft
solution with **INCORRECT** final answer to that problem.  For the
**INCORRECT** draft, walk through the solution step-by-step and check
for any fatal mistakes in the reasoning (for example:  incorrect use
of theorems or facts, incorrect calculations, etc.).  Please note that
once you spot a fatal error in a draft's reasoning, you **MUST** stop
checking the remaining steps of that draft and move on.

For the **INCORRECT** draft, output an overall verdict on the
correctness of the draft using the following tag format:

<overall_verdict>Incorrect</overall_verdict>

While evaluating individual steps, be strict about algebra, logic,
and correct theorem application.  Finally, write a complete, correct
solution (you may reuse sound ideas from the draft, but **DO NOT COPY**
the final answer because it is **INCORRECT**).

Conclude by presenting the final answer only inside \boxed{} (i.e., the
final answer should appear only inside a LaTeX `\boxed{}`).

The problem is as follows:
-- beginning of problem --
{problem}
-- end of problem --

Here is the draft solution with **INCORRECT** final answer you need to
consider:

### Solution:
-- beginning of the draft --
{draft1}
-- end of the draft --

Please:
1.  Carefully evaluate the draft solution step-by-step.  Once you spot
a fatal error in a draft's reasoning, you **MUST** stop checking the
remaining steps of that draft and move on.
2.  Output the correctness verdicts for the draft.
3.  Provide a correct solution to the problem with complete reasoning
steps that lead to the answer.

Remember:  the draft solution has **INCORRECT** final answer.  Please
conclude with the final answer only in \boxed{}.
```

Table 19: Full external-error-signal results for self-drag, where E-ERROR refers to the performance under an explicit external "the draft is incorrect" signal added to the 1F prompt. The absolute pp change relative to DIRECT is annotated as a subscript.

Model	AIME24*		AIME25*		HMMT24*		HMMT25*	
	DIRECT	E-ERROR	DIRECT	E-ERROR	DIRECT	E-ERROR	DIRECT	E-ERROR
GPT-OSS-120B	64.1	32.0 (-32.0)	63.2	39.6 (-23.6)	48.7	30.8 (-17.9)	58.1	35.7 (-22.4)
GPT-OSS-20B	54.4	24.4 (-30.0)	67.0	30.4 (-36.6)	47.4	23.9 (-23.5)	55.5	18.0 (-37.5)
Nemotron-32B	68.8	54.2 (-14.6)	56.2	60.4 (+4.2)	56.9	51.2 (-5.6)	67.0	44.6 (-22.3)
Nemotron-7B	56.2	47.5 (-8.8)	58.0	43.8 (-14.3)	40.6	37.5 (-3.1)	52.2	49.6 (-2.7)
Qwen3-32B	58.6	39.1 (-19.5)	70.5	42.6 (-27.8)	48.1	39.9 (-8.2)	48.9	25.0 (-23.9)
Qwen3-8B	56.2	39.6 (-16.7)	68.8	42.7 (-26.0)	34.4	36.5 (+2.1)	46.9	21.1 (-25.8)
LlamaR1-8B	47.7	27.3 (-20.4)	50.0	31.2 (-18.8)	51.6	21.9 (-29.7)	49.0	31.2 (-17.7)
QwenR1-7B	52.3	26.6 (-25.8)	44.2	37.5 (-6.7)	37.5	21.9 (-15.6)	32.8	12.5 (-20.3)
Gemma4-31B	62.5	53.1 (-9.4)	66.2	65.0 (-1.2)	55.7	67.7 (+12.0)	69.6	67.0 (-2.7)
Gemma4-E4B	56.7	35.7 (-21.0)	56.2	47.4 (-8.9)	34.0	31.9 (-2.1)	51.9	44.4 (-7.5)
Model	GPQA-DIAMOND*		MMLU-REDUX*		CRUXEVAL-I*		GAME OF 24*	
	DIRECT	E-ERROR	DIRECT	E-ERROR	DIRECT	E-ERROR	DIRECT	E-ERROR
GPT-OSS-120B	57.5	42.5 (-14.9)	56.9	67.6 (+10.7)	81.4	92.3 (+10.9)	67.1	41.9 (-25.2)
GPT-OSS-20B	55.0	30.1 (-25.0)	59.9	43.0 (-16.9)	70.4	78.7 (+8.3)	71.0	37.9 (-33.1)
Nemotron-32B	55.9	62.8 (+6.9)	56.9	73.0 (+16.1)	72.5	63.7 (-8.9)	66.2	61.1 (-5.1)
Nemotron-7B	50.1	46.3 (-3.8)	57.2	58.1 (+0.9)	55.2	19.5 (-35.6)	65.3	59.9 (-5.4)
Qwen3-32B	50.3	51.4 (+1.1)	56.2	73.1 (+16.9)	66.5	54.6 (-11.9)	68.5	51.8 (-16.7)
Qwen3-8B	56.1	38.5 (-17.6)	56.8	65.0 (+8.2)	62.8	47.7 (-15.1)	64.7	34.1 (-30.6)
LlamaR1-8B	46.9	30.7 (-16.1)	58.3	48.7 (-9.6)	52.4	43.3 (-9.1)	48.1	15.7 (-32.3)
QwenR1-7B	43.8	13.4 (-30.4)	54.1	42.1 (-12.0)	57.3	37.1 (-20.2)	38.5	26.8 (-11.6)
Gemma4-31B	54.9	71.5 (+16.6)	54.8	85.4 (+30.7)	73.2	67.9 (-5.4)	61.5	57.3 (-4.2)
Gemma4-E4B	50.2	28.6 (-21.5)	56.1	61.5 (+5.4)	63.8	34.0 (-29.7)	66.2	43.9 (-22.3)

Table 20: Full external-error-signal results for cross-drag, where E-ERROR refers to the performance under an explicit external "the draft is incorrect" signal added to the 1F prompt. The absolute pp change relative to DIRECT is annotated as a subscript.

Model	AIME24*		AIME25*		HMMT24*		HMMT25*	
	DIRECT	E-ERROR	DIRECT	E-ERROR	DIRECT	E-ERROR	DIRECT	E-ERROR
GPT-OSS-120B	66.2	25.0 (-41.2)	66.8	34.1 (-32.7)	48.8	26.2 (-22.7)	61.0	40.0 (-21.0)
GPT-OSS-20B	51.9	17.5 (-34.4)	51.9	17.7 (-34.2)	40.2	13.3 (-27.0)	58.6	27.1 (-31.5)
Nemotron-32B	83.8	76.9 (-6.9)	78.4	71.4 (-7.0)	61.7	57.4 (-4.3)	80.4	77.1 (-3.3)
Nemotron-7B	67.5	60.6 (-6.9)	67.8	56.2 (-11.5)	49.2	43.8 (-5.5)	63.1	59.2 (-3.9)
Qwen3-32B	65.0	48.8 (-16.2)	54.8	46.9 (-7.9)	39.1	37.9 (-1.2)	49.7	42.9 (-6.8)
Qwen3-8B	55.6	41.9 (-13.8)	45.7	30.8 (-14.9)	21.9	27.3 (+5.5)	42.3	33.9 (-8.3)
Model	GPQA-DIAMOND*		MMLU-REDUX*		CRUXEVAL-I*		GAME OF 24*	
	DIRECT	E-ERROR	DIRECT	E-ERROR	DIRECT	E-ERROR	DIRECT	E-ERROR
GPT-OSS-120B	58.2	44.1 (-14.1)	67.6	73.8 (+6.2)	89.2	80.6 (-8.6)	77.3	62.6 (-14.7)
GPT-OSS-20B	47.0	32.1 (-14.8)	58.9	50.8 (-8.1)	61.3	48.7 (-12.6)	78.3	74.4 (-3.9)
Nemotron-32B	66.8	64.9 (-1.9)	67.2	80.3 (+13.0)	77.1	68.6 (-8.5)	79.5	74.1 (-5.5)
Nemotron-7B	51.6	50.3 (-1.4)	56.5	64.2 (+7.7)	43.6	39.4 (-4.2)	75.4	69.4 (-6.0)
Qwen3-32B	57.6	54.9 (-2.7)	70.6	80.5 (+9.9)	75.8	67.2 (-8.6)	78.5	60.5 (-17.9)
Qwen3-8B	47.8	38.2 (-9.6)	61.7	66.1 (+4.4)	67.5	51.6 (-15.9)	76.3	48.9 (-27.4)

Table 21: Full self-detected-error-signal results for self-drag, where SD-ERROR refers to the performance on the post-hoc-filtered subset of 1F responses where the model self-detected the draft as incorrect. The number of problems remaining after the post-hoc filtering is shown next to DIRECT.

Model	AIME24*		AIME25*		HMMT24*		HMMT25*	
	DIRECT	SD-ERROR	DIRECT	SD-ERROR	DIRECT	SD-ERROR	DIRECT	SD-ERROR
GPT-OSS-120B	64.1 (8/8)	67.3 (+3.2)	63.2 (9/9)	66.4 (+3.3)	48.7 (14/14)	50.1 (+1.4)	56.6 (16/17)	52.2 (-4.4)
GPT-OSS-20B	54.4 (10/10)	34.9 (-19.4)	67.0 (14/14)	31.6 (-35.4)	47.4 (17/17)	20.7 (-26.7)	53.5 (16/17)	31.8 (-21.7)
Nemotron-32B	68.8 (3/3)	81.1 (+12.4)	40.6 (2/3)	66.9 (+26.3)	56.9 (10/10)	75.4 (+18.6)	67.0 (7/7)	75.7 (+8.8)
Nemotron-7B	56.2 (5/5)	68.0 (+11.8)	53.1 (6/7)	66.2 (+13.0)	40.6 (12/12)	49.1 (+8.5)	53.8 (13/14)	66.1 (+12.2)
Qwen3-32B	58.6 (8/8)	44.0 (-14.6)	71.9 (10/11)	50.1 (-21.7)	48.1 (13/13)	41.5 (-6.6)	46.2 (10/11)	35.6 (-10.7)
Qwen3-8B	56.2 (6/6)	46.0 (-10.3)	71.0 (11/12)	53.2 (-17.8)	34.4 (4/6)	46.4 (+12.1)	46.9 (8/8)	46.9 (+0.0)
LlamaR1-8B	47.7 (19/19)	39.0 (-8.6)	50.0 (9/9)	26.2 (-23.8)	51.6 (8/8)	30.0 (-21.6)	49.0 (6/6)	29.1 (-19.8)
QwenR1-7B	52.3 (16/16)	42.6 (-9.8)	44.8 (12/13)	51.8 (+7.0)	37.5 (8/8)	27.3 (-10.2)	32.4 (11/12)	17.3 (-15.0)
Gemma4-31B	62.5 (4/4)	61.5 (-1.0)	66.2 (5/5)	88.1 (+21.8)	55.7 (12/12)	72.3 (+16.6)	69.8 (6/7)	80.9 (+11.1)
Gemma4-E4B	60.1 (13/14)	38.5 (-21.6)	53.8 (10/12)	44.8 (-8.9)	35.2 (8/9)	34.4 (-0.8)	51.9 (10/10)	53.4 (+1.5)
Model	GPQA-DIAMOND*		MMLU-REDUX*		CRUXEVAL-I*		GAME OF 24*	
	DIRECT	SD-ERROR	DIRECT	SD-ERROR	DIRECT	SD-ERROR	DIRECT	SD-ERROR
GPT-OSS-120B	54.5 (69/92)	46.2 (-8.3)	59.9 (208/301)	76.8 (+17.0)	79.9 (48/369)	70.7 (-9.2)	67.1 (443/520)	78.1 (+11.0)
GPT-OSS-20B	48.9 (101/133)	31.1 (-17.8)	57.6 (425/551)	59.8 (+2.2)	66.5 (289/708)	54.8 (-11.7)	70.8 (448/563)	75.6 (+4.8)
Nemotron-32B	57.9 (56/80)	55.5 (-2.4)	58.9 (94/305)	60.6 (+1.7)	71.1 (76/102)	74.8 (+3.8)	66.2 (310/311)	62.5 (-3.8)
Nemotron-7B	51.6 (84/106)	37.6 (-14.1)	57.6 (217/593)	50.3 (-7.3)	52.7 (273/712)	43.7 (-8.9)	65.2 (403/405)	68.4 (+3.2)
Qwen3-32B	51.4 (65/72)	34.5 (-16.9)	56.0 (145/232)	72.6 (+16.6)	67.0 (99/108)	63.4 (-3.6)	68.5 (381/391)	61.5 (-7.0)
Qwen3-8B	56.6 (61/73)	28.2 (-28.3)	58.8 (155/287)	56.9 (-1.9)	64.4 (94/109)	54.9 (-9.5)	63.9 (290/351)	62.0 (-1.9)
LlamaR1-8B	47.1 (128/137)	32.9 (-14.1)	56.4 (1084/1496)	54.6 (-1.8)	50.4 (435/497)	57.0 (+6.6)	49.4 (591/789)	39.5 (-9.9)
QwenR1-7B	46.8 (65/76)	20.0 (-26.8)	50.7 (1013/1489)	47.4 (-3.3)	56.8 (362/412)	58.0 (+1.2)	36.5 (70/98)	47.4 (+10.8)
Gemma4-31B	55.2 (37/41)	70.9 (+15.6)	56.2 (55/100)	78.2 (+22.0)	71.4 (12/14)	82.6 (+11.2)	61.5 (12/12)	52.0 (-9.4)
Gemma4-E4B	50.0 (70/105)	27.7 (-22.3)	62.0 (161/492)	64.2 (+2.2)	64.0 (169/337)	55.7 (-8.3)	66.7 (175/182)	58.0 (-8.7)

Table 22: Full self-detected-error-signal results for cross-drag, where SD-ERROR refers to the performance on the post-hoc-filtered subset of 1F responses where the model self-detected the draft as incorrect. The number of problems remaining after the post-hoc filtering is shown next to DIRECT.

Model	AIME24*		AIME25*		HMMT24*		HMMT25*	
	DIRECT	SD-ERROR	DIRECT	SD-ERROR	DIRECT	SD-ERROR	DIRECT	SD-ERROR
GPT-OSS-120B	66.2 (10/10)	53.6 (-12.7)	66.8 (13/13)	70.9 (+4.0)	48.8 (16/16)	52.9 (+4.1)	60.5 (19/21)	64.3 (+3.7)
GPT-OSS-20B	51.9 (10/10)	28.5 (-23.4)	51.9 (13/13)	43.3 (-8.6)	40.2 (16/16)	16.4 (-23.8)	57.2 (20/21)	33.9 (-23.3)
Nemotron-32B	83.8 (10/10)	93.3 (+9.5)	78.4 (13/13)	80.6 (+2.3)	61.7 (16/16)	66.2 (+4.5)	80.4 (21/21)	85.7 (+5.3)
Nemotron-7B	67.5 (10/10)	76.0 (+8.5)	67.8 (13/13)	83.1 (+15.3)	50.8 (15/16)	54.4 (+3.6)	63.1 (21/21)	72.1 (+9.0)
Qwen3-32B	65.0 (10/10)	49.0 (-16.0)	55.7 (12/13)	53.7 (-2.1)	39.1 (16/16)	47.3 (+8.2)	50.3 (20/21)	51.4 (+1.1)
Qwen3-8B	61.8 (9/10)	54.1 (-7.7)	50.0 (8/13)	44.3 (-5.7)	20.1 (14/16)	36.2 (+16.1)	46.5 (18/21)	46.3 (-0.2)
Model	GPQA-DIAMOND*		MMLU-REDUX*		CRUXEVAL-I*		GAME OF 24*	
	DIRECT	SD-ERROR	DIRECT	SD-ERROR	DIRECT	SD-ERROR	DIRECT	SD-ERROR
GPT-OSS-120B	57.9 (112/132)	48.9 (-9.1)	73.0 (450/569)	79.7 (+6.7)	90.6 (173/188)	85.7 (-4.9)	77.5 (634/653)	80.8 (+3.4)
GPT-OSS-20B	44.8 (115/132)	25.0 (-19.8)	61.4 (458/569)	61.3 (-0.1)	61.5 (175/188)	53.5 (-8.0)	78.5 (603/653)	79.0 (+0.5)
Nemotron-32B	66.5 (104/132)	66.0 (-0.5)	77.1 (293/569)	74.2 (-2.8)	79.1 (164/188)	74.7 (-4.4)	79.4 (649/653)	76.0 (-3.4)
Nemotron-7B	50.8 (103/132)	56.5 (+5.7)	62.7 (266/569)	59.9 (-2.8)	44.4 (163/188)	52.4 (+8.0)	75.6 (641/653)	76.8 (+1.3)
Qwen3-32B	55.3 (110/132)	43.3 (-12.1)	74.6 (426/569)	77.4 (+2.9)	75.5 (161/188)	72.5 (-3.0)	78.3 (645/653)	70.3 (-8.0)
Qwen3-8B	44.5 (104/132)	32.3 (-12.2)	70.4 (332/569)	67.9 (-2.4)	67.3 (158/188)	60.1 (-7.2)	76.2 (576/653)	74.7 (-1.5)

F Additional Details of Context Denoising Experiments

F.1 Implementation of Revise and Filter

We provide the full implementation details of both methods below. We note that both involve multiple inference steps and when using the output from a previous step, we always remove the thinking traces wrapped in `<think></think>` for manageable context length. We only present the prompt templates for 1F for illustration.

Revise consists of 2 steps of inference:

(revise). given the **`problem`** and one **`draft`**, the model is asked to address any mistakes and produce a revised solution. Denote the output as the **`revised_draft`**. This inference needs to be repeated for each draft independently.

Revise *(revise step,* 1F*)*

```
You are given a problem and one draft solution to that problem.  Your
task is to revise the draft to produce an improved, complete solution
that addresses any mistakes or shortcomings in the original draft.
Ensure that your revised solution includes all necessary reasoning
steps leading to the final answer and conclude by presenting the final
answer only inside \boxed{}.

The problem is as follows:
-- beginning of problem --
{problem}
-- end of problem --

Here is the draft solution you need to revise:
-- beginning of the draft --
{draft1}
-- end of the draft --

Remember:  conclude with the final answer only in \boxed{}.
```

(solve). given the **`problem`** and the **`revised_draft`**, the model is asked to generate a new solution. The prompt template is the same as the VER ablation template except that the draft in the context is replaced with its revised version. We note that the instruction to check the draft correctness is removed to optimize model performance.

Revise *(solve step,* 1F*)*

```
In this task, you will be provided with a problem and one draft
solution to that problem.  If the draft is incorrect, write a complete,
correct solution (you may reuse sound ideas from the draft).  If the
draft is correct, you may adopt and polish it as the final solution.

Conclude by presenting the final answer only inside \boxed{} (i.e., the
final answer should appear only inside a LaTeX `\boxed{}`).

The problem is as follows:
-- beginning of problem --
{problem}
-- end of problem --

Here is the draft solution you need to consider:

### Solution:
-- beginning of the draft --
{revised_draft1}
-- end of the draft --
```

```
Please provide a correct solution to the problem with complete
reasoning steps that lead to the answer.

Remember:  conclude with the final answer only in \boxed{}.
```

FILTER consists of 3 steps of inference:

(strategy). given the **`problem`**, the model is asked to produce a high-level strategy to solve the problem. The model is not allowed to include any intermediate steps or the final answer. Denote the output as the **`strategy`**.

FILTER *(strategy stage, single draft)*

```
You are given a problem.  Your task is **NOT to solve the problem**.
Instead, produce a **high-level strategy or a plan** that you would use
to approach the problem.

Requirements for your response:
1.  **Do NOT give any step-by-step computation or any final answers**.
2.  Produce a concise, structured plan (use numbered steps or short
bullet points) of you would take as if you would attempt to solve the
given question.  You can include any useful heuristics, ways to reduce
problem complexity, and common pitfalls to avoid.
3.  Finally, include a short checklist to verify a candidate answer.

Now, produce the high-level strategy (no solution) for the following
problem:
-- beginning of problem --
{problem}
-- end of problem --

Remember that you **do not attempt to solve the question or include any
intermediate steps or the final answer in your response**.
```

(filter). given the **`problem`**, the **`strategy`**, and one **`draft`**, the model is asked to filter the draft to only keep the correct and useful steps toward solving the problem according to the **`strategy`** it produced in the previous step. Similar to the revise step in REVISE, this step needs to be repeated for each draft independently. Denote the output as the **`filtered_draft`**.

FILTER *(filter step,* 1F*)*

```
You are given a problem, a draft solution attempt, and a high-level
strategy for solving the problem.  Your task is to identify the useful
components or intermediate steps in the draft solution that align
with the given strategy.  Focus on selecting steps that contribute
meaningfully to progressing toward the final solution, even if the
draft solution as a whole may contain mistakes or irrelevant parts.
Format your response as numbered steps, followed by a concise (no more
than 2 sentences), clear explanation of how each step fits into the
strategy.  Remember:  **only include the useful and correct parts** and
**do not include any incorrect or irrelevant parts** in your response.

The problem is as following:
-- beginning of problem --
{problem}
-- end of problem --

You should consider the following strategy to solve the problem:
-- beginning of strategy --
{strategy}
-- end of strategy --
```

```
Please **only output the useful and correct components or intermediate
steps that align with the above strategy** from the following draft
solution:
-- beginning of draft solution --
{draft1}
-- end of draft solution --

Remember to format your response as numbered steps, followed by a
concise explanation of why they are useful.
```

(solve). given the **problem** and the **filtered_draft**, the model is asked to generate a new solution. Similar to REVISE, the prompt template used at this step is the same as the 1F template in Section A.1 except that (1) the draft in the context is replaced with its revised version and (2) the instruction to check the correctness of the draft is removed to optimize model performance.

FILTER *(solve step,* 1F*)*

```
In this task, you will be provided with a problem and one draft
solution to that problem.  If the draft is incorrect, write a complete,
correct solution (you may reuse sound ideas from the draft).  If the
draft is correct, you may adopt and polish it as the final solution.

Conclude by presenting the final answer only inside \boxed{} (i.e., the
final answer should appear only inside a LaTeX `\boxed{}`).

The problem is as follows:
-- beginning of problem --
{problem}
-- end of problem --

Here is the draft solution you need to consider:

### Solution:
-- beginning of the draft --
{filtered_draft1}
-- end of the draft --

Please provide a correct solution to the problem with complete
reasoning steps that lead to the answer.

Remember:  conclude with the final answer only in \boxed{}.
```

F.2 ADDITIONAL RESULTS

Here we provide the full per-task results of the two context denoising approaches discussed in Section 4.1 in the self-drag setting. Table 23 reports DIRECT, 1F, FILTER, and REVISE pass@1 across all 10 models and 8 tasks. The cross-drag analogues are reported in Section B.

F.3 OUTCOME-BUCKET DECOMPOSITION OF MITIGATION

Figure 16 decomposes the 1F-to-mitigation transition into outcome buckets (e.g., *recovered*, *preserved*, *regressed*), macro-averaged over the eight tasks. FILTER and REVISE are plotted as independent mitigations.

F.4 RECURSIVE FILTER: PER-ROUND RAW STATISTICS

Table 25 reports the per-round pass@1 statistics underlying Figure 3. Round 0 is the clean-slate initial generation; subsequent rounds reuse the previous round's response, either raw (standard recursive) or FILTER-denoised.

Table 23: **Full context-denoising results for self-drag** (pass@1, mean ± std across trajectories, in pp). REVISE and FILTER are applied to the same in-context draft used for 1F. DIRECT and 1F are reproduced for reference. Gemma4 REVISE/FILTER cells are reported without std.

	AIME24*				AIME25*			
	DIRECT	REVISE	FILTER	1F	DIRECT	REVISE	FILTER	1F
GPT-OSS-120B	64.06 ±13.89	43.75 ±9.88	35.94 ±10.25	29.69 ±16.16	63.19 ±14.55	76.39 ±7.73	54.86 ±12.86	57.64 ±15.45
GPT-OSS-20B	54.37 ±10.59	56.88 ±7.04	55.00 ±6.12	26.25 ±9.27	66.96 ±12.60	67.86 ±4.72	70.09 ±6.17	24.11 ±10.83
Nemotron-32B	68.75 ±21.95	45.83 ±11.02	50.00 ±11.79	45.83 ±16.14	56.25 ±28.18	62.50 ±11.02	62.50 ±16.14	35.42 ±15.45
Nemotron-7B	56.25 ±16.15	31.25 ±12.69	63.75 ±6.96	36.25 ±15.76	58.04 ±15.54	57.14 ±5.05	50.89 ±6.62	37.50 ±19.48
Qwen3-32B	58.59 ±17.52	39.06 ±2.71	50.78 ±12.67	28.91 ±6.94	70.45 ±13.82	62.50 ±6.72	44.32 ±6.72	30.68 ±6.33
Qwen3-8B	56.25 ±19.43	28.12 ±5.80	30.21 ±10.97	23.96 ±6.51	68.75 ±10.83	67.71 ±5.41	64.58 ±9.99	41.67 ±8.59
LlamaR1-8B	47.70 ±11.54	50.00 ±4.36	49.34 ±5.22	31.25 ±9.25	50.00 ±13.03	26.39 ±6.05	37.50 ±8.67	20.14 ±8.30
QwenR1-7B	52.34 ±13.95	37.89 ±5.05	46.88 ±7.97	30.47 ±6.58	44.23 ±12.61	41.35 ±6.31	42.79 ±4.48	30.77 ±8.38
Gemma4-31B	62.50 ±21.65	75.00	75.00	43.75 ±15.31	66.25 ±18.33	100.00	100.00	63.75 ±13.17
Gemma4-E4B	56.70 ±16.46	28.57	28.57	35.27 ±4.87	56.25 ±12.67	41.67	41.67	39.06 ±7.49
	HMMT24*				HMMT25*			
	DIRECT	REVISE	FILTER	1F	DIRECT	REVISE	FILTER	1F
GPT-OSS-120B	48.66 ±10.78	50.45 ±5.19	42.86 ±7.99	33.04 ±3.89	58.09 ±8.02	55.88 ±2.08	48.16 ±9.75	40.44 ±5.84
GPT-OSS-20B	47.43 ±8.44	59.56 ±7.01	47.06 ±5.09	19.12 ±5.50	55.51 ±9.29	57.35 ±3.60	46.32 ±3.82	17.65 ±5.88
Nemotron-32B	56.88 ±16.09	53.12 ±3.48	58.75 ±5.45	41.88 ±4.96	66.96 ±12.08	41.07 ±6.92	56.25 ±6.62	46.43 ±7.99
Nemotron-7B	40.62 ±12.10	41.67 ±6.91	42.71 ±5.00	33.85 ±5.68	52.23 ±13.30	50.89 ±3.46	47.32 ±3.89	41.07 ±3.57
Qwen3-32B	48.08 ±13.46	47.60 ±3.30	38.46 ±5.77	23.56 ±4.88	48.86 ±14.33	26.14 ±4.40	32.95 ±5.45	13.07 ±4.79
Qwen3-8B	34.37 ±17.15	37.50 ±7.22	44.79 ±9.26	33.33 ±8.33	46.88 ±12.88	36.72 ±6.58	32.03 ±7.93	24.22 ±4.88
LlamaR1-8B	51.56 ±13.17	30.47 ±4.88	35.94 ±5.18	25.78 ±13.05	48.96 ±18.13	38.54 ±8.27	43.75 ±13.01	25.00 ±13.18
QwenR1-7B	37.50 ±13.98	35.16 ±7.61	35.16 ±7.61	21.88 ±6.99	32.81 ±11.96	22.40 ±3.57	29.17 ±6.91	13.54 ±5.80
Gemma4-31B	55.73 ±11.30	75.00	58.33	59.38 ±7.14	69.64 ±14.17	85.71	85.71	62.50 ±9.94
Gemma4-E4B	34.03 ±16.42	11.11	33.33	22.22 ±9.62	51.87 ±13.79	20.00	10.00	38.12 ±4.96
	GPQA-DIAMOND*				MMLU-REDUX*			
	DIRECT	REVISE	FILTER	1F	DIRECT	REVISE	FILTER	1F
GPT-OSS-120B	57.47 ±4.35	43.07 ±2.46	46.06 ±2.27	31.66 ±3.26	56.89 ±1.92	36.79 ±1.10	33.68 ±2.06	28.92 ±1.74
GPT-OSS-20B	55.03 ±3.10	42.25 ±2.35	48.31 ±1.49	23.68 ±2.93	59.89 ±1.75	39.95 ±0.84	46.50 ±1.08	26.70 ±0.76
Nemotron-32B	55.94 ±2.91	20.16 ±0.75	55.31 ±2.10	24.84 ±3.43	56.91 ±1.95	8.87 ±0.26	43.18 ±0.86	13.34 ±0.99
Nemotron-7B	50.06 ±3.36	18.04 ±1.76	39.80 ±1.41	20.58 ±2.89	57.19 ±1.05	10.02 ±0.21	39.09 ±0.42	13.64 ±0.57
Qwen3-32B	50.26 ±4.26	12.93 ±1.70	16.41 ±1.59	19.27 ±2.13	56.25 ±1.89	9.24 ±0.49	11.45 ±0.71	23.03 ±1.58
Qwen3-8B	56.08 ±5.11	13.87 ±2.39	26.37 ±1.94	16.87 ±1.41	56.84 ±2.91	10.43 ±0.28	18.82 ±1.22	14.00 ±0.49
LlamaR1-8B	46.85 ±3.15	25.64 ±1.19	43.57 ±1.00	23.27 ±0.85	58.35 ±1.18	45.72 ±0.48	59.70 ±0.32	44.82 ±1.05
QwenR1-7B	43.83 ±2.82	17.35 ±1.14	22.78 ±2.23	17.60 ±2.18	54.05 ±1.25	50.60 ±0.59	58.79 ±0.41	44.15 ±0.84
Gemma4-31B	54.88 ±7.27	31.71	21.95	35.67 ±1.90	54.75 ±4.93	13.00	3.00	26.00 ±2.60
Gemma4-E4B	50.18 ±3.98	5.71	18.10	9.76 ±1.81	56.15 ±1.62	6.90	8.52	10.81 ±0.64
	CRUXEVAL-I*				GAME OF 24*			
	DIRECT	REVISE	FILTER	1F	DIRECT	REVISE	FILTER	1F
GPT-OSS-120B	81.40 ±7.52	85.20 ±1.07	89.83 ±1.08	93.41 ±1.86	67.14 ±2.24	71.81 ±0.31	32.30 ±1.69	49.82 ±2.11
GPT-OSS-20B	70.37 ±13.19	72.31 ±6.35	68.33 ±5.24	78.24 ±3.47	70.97 ±2.75	53.57 ±1.68	35.92 ±2.02	32.79 ±5.88
Nemotron-32B	72.55 ±2.79	50.67 ±1.32	53.85 ±3.02	40.69 ±1.98	66.18 ±2.73	69.90 ±0.81	59.57 ±1.38	47.43 ±1.98
Nemotron-7B	55.15 ±6.73	23.84 ±0.61	20.83 ±1.84	12.55 ±0.53	65.31 ±2.19	69.98 ±0.46	55.20 ±1.93	43.61 ±1.12
Qwen3-32B	66.49 ±4.03	49.03 ±1.14	44.88 ±3.04	40.39 ±1.53	68.48 ±2.33	45.27 ±1.32	41.24 ±1.88	32.82 ±1.64
Qwen3-8B	62.84 ±4.94	31.25 ±0.95	38.13 ±2.36	30.22 ±1.13	64.67 ±3.10	29.70 ±0.68	44.82 ±0.90	20.53 ±2.22
LlamaR1-8B	52.35 ±1.73	37.07 ±0.57	48.86 ±0.91	39.95 ±1.62	48.05 ±1.71	21.38 ±0.80	35.91 ±1.48	9.09 ±1.17
QwenR1-7B	57.28 ±2.03	28.68 ±0.78	38.69 ±0.83	27.00 ±1.28	38.46 ±4.19	28.70 ±0.95	35.01 ±2.52	15.82 ±1.71
Gemma4-31B	73.21 ±8.18	53.12 ±4.16	35.27 ±5.19	43.75 ±7.09	61.46 ±9.72	68.75 ±4.66	78.12 ±7.44	47.40 ±4.63
Gemma4-E4B	63.76 ±2.89	21.99 ±0.61	44.86 ±1.49	17.86 ±1.39	66.21 ±3.31	33.17 ±0.33	29.57 ±2.65	36.09 ±1.52

Table 24: **Per-model recovery of FILTER and REVISE against self-drag.** For each (model, method), we define Recovery $= (\mathrm{Acc}_{\mathrm{method}} - \mathrm{Acc}_{\mathrm{1F}})/(\mathrm{Acc}_{\mathrm{DIRECT}} - \mathrm{Acc}_{\mathrm{1F}})$, averaged over the eight tasks; 100% fully closes the 1F→DIRECT gap and negative values indicate further regression. Recovery is normalized by the gap: 100% means matching DIRECT and is not a ceiling. FILTER achieves positive recovery on every model, while REVISE peaks on stronger reasoners but regresses on Gemma4-E4B—mirroring Figure 6 (see Section 4.1).

	GPT-OSS		Nemotron		Qwen3		Gemma4		R1-Distill	
	120B	20B	32B	7B	32B	8B	31B	E4B	8B	7B
FILTER	15.7%	73.8%	67.0%	61.5%	26.1%	39.6%	36.1%	3.6%	69.4%	61.7%
REVISE	75.1%	93.1%	24.4%	31.0%	29.2%	21.0%	72.7%	−17.1%	28.8%	35.8%

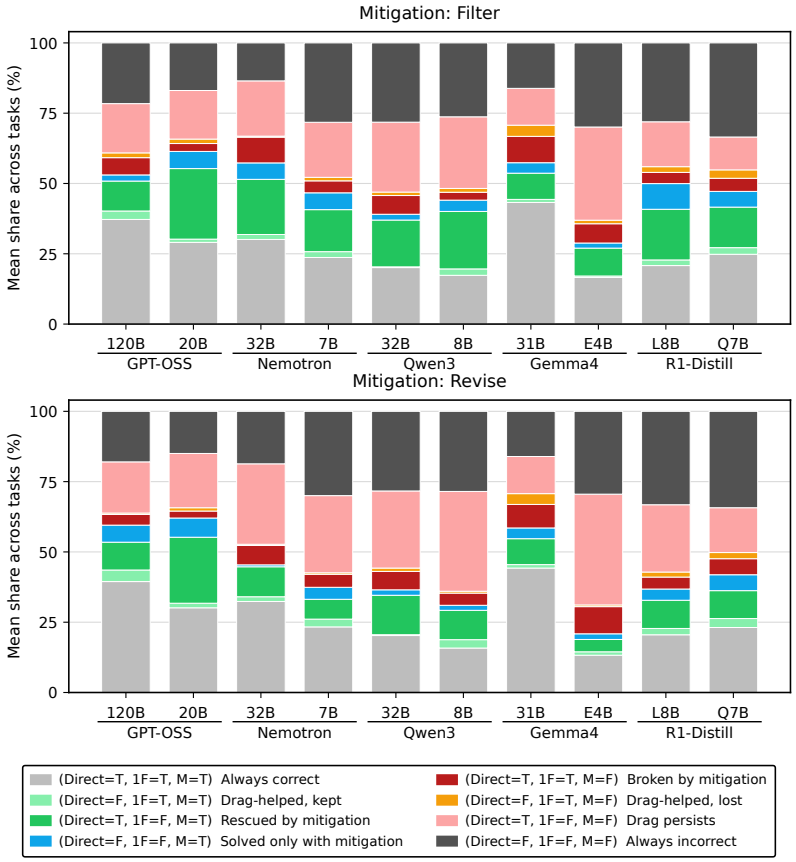

Figure 16: **Outcome-bucket decomposition of context denoising** (macro-averaged across tasks). Each panel shows how FILTER or REVISE redistributes per-problem outcomes relative to 1F and DIRECT. The margin of improvement can be considered as the height of green and blue segments subtracted by the red and yellow segments.

Table 25: Raw per-round statistics (pass@1, mean ± std, in %). Round 0 is the clean-slate initial generation mirrored onto the standard recursive curve. The parenthetical on each w/ recursive FILTER cell is the absolute pp delta relative to the same-round Iterative Refinement baseline.

	AIME24		AIME25	
Round	Iter. Refinement	w/ FILTER	Iter. Refinement	w/ FILTER
0	76.04 ±5.92	76.04 ±5.92	71.88 ±5.00	71.88 ±5.00
1	72.77 ±3.46	78.75 ±3.51 (+5.98)	68.48 ±3.39	75.62 ±5.36 (+7.15)
2	69.87 ±3.47	79.58 ±4.84 (+9.71)	64.14 ±3.29	77.87 ±5.11 (+13.73)
3	68.33 ±3.94	80.21 ±5.06 (+11.88)	61.59 ±4.23	79.74 ±4.82 (+18.15)
4	67.25 ±2.67	81.46 ±4.56 (+14.21)	60.00 ±3.62	77.67 ±4.94 (+17.67)
5	67.07 ±3.88	83.75 ±4.23 (+16.68)	59.83 ±3.72	75.53 ±4.77 (+15.70)
6	66.17 ±3.24	80.83 ±4.79 (+14.66)	59.53 ±4.20	78.25 ±5.25 (+18.72)
7	64.90 ±2.83	80.63 ±5.68 (+15.73)	59.47 ±4.63	77.82 ±5.15 (+18.35)
8	64.35 ±3.47	82.46 ±4.48 (+18.12)	57.34 ±5.17	79.48 ±4.48 (+22.14)
9	63.54 ±3.28	82.05 ±4.03 (+18.51)	57.41 ±6.11	78.25 ±3.50 (+20.84)
10	63.53 ±3.58	80.59 ±3.74 (+17.06)	57.89 ±4.79	78.46 ±3.85 (+20.57)
11	64.40 ±3.05	81.85 ±4.53 (+17.45)	57.47 ±4.95	78.67 ±4.34 (+21.20)
12	63.27 ±3.24	83.29 ±3.38 (+20.02)	57.96 ±4.24	80.11 ±3.93 (+22.15)
13	63.16 ±2.99	82.26 ±4.67 (+19.09)	58.15 ±4.49	78.86 ±3.70 (+20.71)
14	62.75 ±2.93	82.05 ±4.05 (+19.30)	57.58 ±4.90	77.61 ±4.54 (+20.04)
15	63.54 ±3.62	83.49 ±5.77 (+19.95)	58.01 ±5.04	77.84 ±6.16 (+19.84)

	HMMT24		HMMT25	
Round	Iter. Refinement	w/ FILTER	Iter. Refinement	w/ FILTER
0	48.75 ±7.16	48.75 ±7.16	55.42 ±4.39	55.42 ±4.39
1	46.74 ±3.70	53.96 ±6.99 (+7.21)	48.03 ±3.10	58.75 ±4.70 (+10.72)
2	43.09 ±4.48	56.57 ±5.41 (+13.49)	44.23 ±4.23	57.71 ±4.03 (+13.48)
3	39.64 ±3.51	57.63 ±4.59 (+17.99)	41.20 ±4.18	58.96 ±4.67 (+17.76)
4	39.34 ±3.76	59.19 ±5.16 (+19.85)	40.44 ±3.35	59.17 ±4.79 (+18.72)
5	38.05 ±3.84	57.54 ±6.71 (+19.49)	40.85 ±4.06	60.62 ±5.43 (+19.78)
6	37.66 ±3.83	56.68 ±6.48 (+19.02)	39.78 ±3.40	59.79 ±5.33 (+20.01)
7	36.82 ±3.43	58.79 ±4.75 (+21.97)	39.18 ±2.64	62.08 ±4.70 (+22.90)
8	37.24 ±3.16	58.97 ±4.69 (+21.73)	39.34 ±3.50	63.27 ±4.92 (+23.93)
9	37.37 ±3.40	59.23 ±7.63 (+21.86)	38.37 ±3.18	59.92 ±4.03 (+21.55)
10	36.63 ±3.16	59.83 ±5.17 (+23.20)	39.53 ±4.15	61.57 ±6.21 (+22.04)
11	35.76 ±3.14	58.36 ±6.48 (+22.61)	39.39 ±5.38	62.19 ±5.67 (+22.80)
12	36.58 ±3.59	57.11 ±4.10 (+20.54)	38.01 ±5.28	60.32 ±6.67 (+22.31)
13	35.20 ±2.77	57.68 ±6.91 (+22.48)	37.57 ±4.82	62.16 ±8.90 (+24.60)
14	36.02 ±3.54	58.66 ±6.79 (+22.65)	39.01 ±4.15	61.29 ±5.34 (+22.28)
15	36.51 ±2.76	58.28 ±4.54 (+21.77)	39.22 ±4.23	59.82 ±4.37 (+20.60)

F.5 Iterative Refinement and Recursive Filtering Beyond GPT-OSS-20B

We run the naive iterative-refinement protocol of Section 2.2 (unfiltered, rounds 0–8, 8 independent chains) on four additional models, averaging over AIME24 and AIME25. Table 26 shows that self-deterioration is not universal: outcomes range from clear deterioration (GPT-OSS-20B, -11.3 points by round 8), through roughly flat trajectories (Qwen3-32B and Qwen3-8B), to mild (Nemotron-32B, $+2.9$) and clear self-improvement (Gemma4-31B, $+9.4$). The ordering of the five models by iterative outcome matches their ordering by single-step drag on the same benchmarks, and the two models that improve are also those that recover under correct self-verification (Figure 5, right). With only five models, we report this as an empirical association rather than a predictive relationship. The GPT-OSS-20B row comes from this 8-chain run on AIME24/25 and is separate from the 16-trajectory run in Table 25, so the two round-8 changes differ slightly.

Table 26: **Naive iterative refinement across models** (AIME24 and AIME25, mean over the two benchmarks and 8 independent chains). The single-step drag column is the mean 1F−DIRECT change on the same two benchmarks (Table 3); the last column is the round-8 minus round-0 change under unfiltered iterative refinement.

Model	Single-step drag (1F−DIRECT)	Iterative change (round $8-0$)
GPT-OSS-20B	-35.48	-11.3
Qwen3-32B	-34.73	-3.5
Qwen3-8B	-29.68	-0.6
Nemotron-32B	-21.88	+2.9
Gemma4-31B	-10.62	+9.4

Single-step drag alone does not determine iterative outcomes; they also depend on how reliably the model retains correct drafts (1T). GPT-OSS-20B and Qwen3-32B have nearly identical single-step drag (-35.5 vs. -34.7) but different trajectories (-11.3 vs. -3.5), consistent with differences in retention; a per-model decomposition is left to future work.

We also apply recursive FILTER to Qwen3-32B, whose unfiltered trajectory declines mildly (Table 27). Filtering reverses the trend: averaged over the four math benchmarks, it reaches 66.5% at round 8, 4.2 points above unfiltered refinement (62.3%) and 2.1 points above the round-0 baseline, consistent with the GPT-OSS-20B result in Figure 3.

Table 27: **Recursive filtering on Qwen3-32B** (pass@1, %, mean over AIME24, AIME25, HMMT24 and HMMT25). Round 0 is the shared clean-slate generation; parentheses give the gain of recursive filtering over unfiltered iterative refinement at the same round.

Round	0	1	2	3	4	5	6	7	8
Iter. refinement	64.38	63.96	64.06	62.60	62.40	62.50	63.85	63.75	62.29
w/ recursive FILTER	64.38	66.15 (+2.19)	65.79 (+1.73)	66.41 (+3.81)	66.41 (+4.01)	66.09 (+3.59)	66.20 (+2.35)	65.89 (+2.14)	66.47 (+4.18)

F.6 Qualitative Case Study: Filter vs. Revise on the Same Draft

We examine one instance from the recoverable-drag bucket of Figure 16: AIME25 problem #25 for GPT-OSS-20B, with the same problem and the same flawed draft given to both methods. Its pass rate is 0.69 under DIRECT, drops to 0.25 under 1F, and rises to 1.00 under either FILTER or REVISE. The problem asks, for the vertices of a regular 24-gon, for the number of ways to draw 12 segments of equal length so that each vertex is an endpoint of exactly one segment. The quotations below are excerpts from the model outputs.

The draft. The draft gets the structure right: equal-length chords correspond to a fixed step k, the resulting graph splits into cycles of length $m = 24/\gcd(24, k)$, and an odd cycle admits no perfect matching. It then makes a fatal error, claiming that an even cycle C_m has $2^{m/2-1}$ perfect matchings "since we can choose either of the two alternating edge sets." That justification gives 2, not $2^{m/2-1}$, and the error inflates every row of the draft's table.

How the drag shows up under 1F. The verifier does not overlook the draft; it re-derives all twelve rows using the draft's own formula ("$(2^{6-1})^2 = \ldots = 1024$. Draft says 1024. Good."). Its effort goes into checking what the draft got right, and the flawed premise is never tested.

FILTER. The draft-blind strategy states independently that "a cycle of even length has exactly two perfect matchings." This gives the filter step a reference that does not depend on the draft. The filter flags the formula as wrong for $m > 4$ and drops it together with the entire numeric table. Because FILTER never passes on a final answer, the solver receives only the rules and their conditions.

REVISE. REVISE reaches the same diagnosis ("For $m = 6$: $2^{3-1} = 4$. But earlier we found only 2 matchings for C_6") but edits in place: it keeps the table's shape, changes the matchings column and the total, restates the corrected lemma with checkable reasoning, and verifies it on a smaller polygon.

Takeaway. Both methods succeed because the solver no longer sees a draft that vouches for its own flawed answer; it sees material it must check and recompute. FILTER gets there by removal, keeping only answer-free steps that agree with an independent strategy; REVISE gets there by repairing the draft in place.

F.7 RELATION TO PRIOR SELF-IMPROVEMENT STRATEGIES

The conditions of Sections 2 and 3 and the two denoising methods of Section 4.1 cover six common strategies for handling prior attempts, each adapted to our single-turn setting with models, prompts, subsets and verifiers held fixed: reusing raw attempts (Wang et al., 2024; Huang et al., 2024) (1F, Section 2.2); labeling them externally (Stechly et al., 2023; Tyen et al., 2024) (Section 3.1); verifying first (Madaan et al., 2023; Huang et al., 2024) (the 1F verification step); repairing before reuse (REVISE); selective reuse (Zhao et al., 2024; Shinn et al., 2023) (FILTER, a minimal model-internal instance); and discarding and resampling (Wang et al., 2023) (DIRECT with maj@k, Section A.6). Labels and verification do not reset influence, and repair helps but inherits it. Discarding forfeits the gains of correct drafts (Section C.4) and is unavailable when context arrives externally, while selective reuse gives the most consistent single-step gains and flips iterative deterioration. Absolute numbers from the original papers are not comparable, since our metrics are computed on per-model reasonably-hard subsets.

G CONDITIONAL FALLBACK TRAINING FOR CONTEXTUAL DRAG

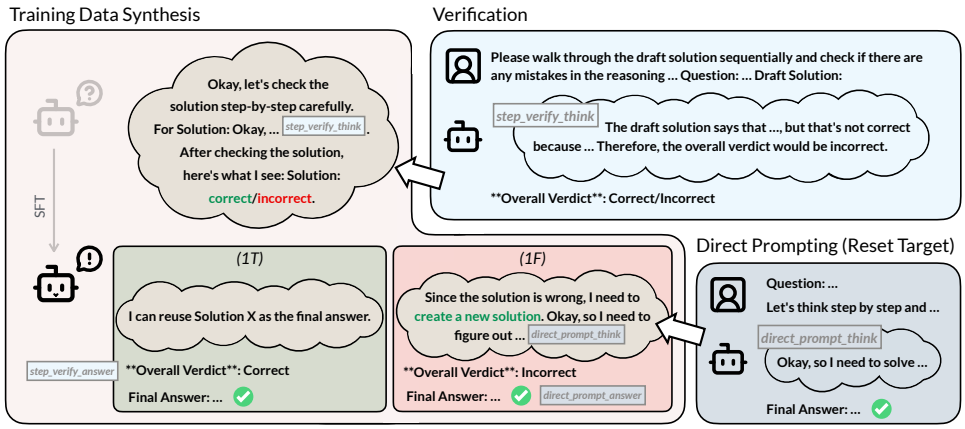

Figure 17: **Overview of the proposed training procedure**: Synthesized reasoning traces teach the model to verify in-context drafts before solving. If the model identifies the draft as incorrect, as in the case of 1F shown above, it is trained to ignore the draft and revert to a reasoning trajectory sampled from clean-slate generation. If the model verifies the draft to be correct, it is trained to reuse the answer in the draft.

G.1 OVERVIEW

Motivated by our finding that contextual drag persists even after correct recognition (Section 3), we explore a training-time intervention: a *conditional fallback training* recipe via supervised fine-tuning (SFT) that teaches the model an explicit *context reset*. Concretely, after step-by-step verification, the model either reuses a draft that is verified as correct, or switches to a clean-slate solution trajectory when it detects an error in the draft's reasoning. We instantiate this on GPT-OSS-20B, the model in our suite that exhibits the largest degradation under 1F even after error signals. The remainder of this appendix documents data curation and trace construction (Section G.2), training (Section G.3), 1F/1T results (Section G.4), and a comparison to outcome-based RL (Section G.5). Throughout, we adopt *context reset* as the canonical term for the post-verification switch to a clean-slate trajectory.

Headline finding. Conditional fallback training is not an effective mitigation. It improves 1F pass@1 on the four hard-math benchmarks (Table 29), but incurs a 22–28 pp absolute drop on 1T pass@1 on the same benchmarks (Table 30), a larger cost on average than the 1F gain, and 1F remains below DIRECT. We report it as evidence that contextual drag is not easily trained away, and that inference-time context control (Section 4.1) is the more effective mitigation.

G.2 DATA CURATION: QUESTION SELECTION FROM BIGMATHRL

To obtain a set of questions on which we can achieve reliable verification, we draw problems from BigMathRL Albalak et al. (2025) and restrict to a reasonably hard subset using an external solver filter: we retain only questions whose solve rate under Llama3.1-8B (provided in the dataset) is below 0.5. From this filtered pool, we subsample 40,000 questions with a fixed source mixture across the underlying BigMathRL data sources.

We report the data composition of the question pool in Table 28. We conducted balanced subsampling such that for small sources, we include the full available set.

For each selected question, we sample candidate draft solutions from a family of open-weight reasoning models: GPT-OSS-20B (with medium thinking effort), Qwen3-8B (with thinking), OpenReasoning-Nemotron-7B, LlamaR1-8B, and QwenR1-7B. Concretely, we sample 4 independent DIRECT responses per (question, model) pair, yielding up to 20 drafts per question. We then determine draft correctness using the `math_verify` package Kydlíček (2024), which provides verifiable answer checking for math problems.

Table 28: Source mixture of the 40k question subset used for synthesis. "Available" denotes the number of filtered questions per source after applying the Llama 3.1 8B solve-rate < 0.5 criterion.

Source	Available	Sampled	Sample rate
cn_k12	37,228	6,729	18.1%
orca_math	27,153	6,728	24.8%
olympiads	29,941	6,728	22.5%
aops_forum	5,067	5,067	100.0%
gsm8k	159	159	100.0%
amc_aime	71	71	100.0%
math	2,780	2,780	100.0%
omnimath	2,033	2,033	100.0%
openmath	281	281	100.0%
harp	2,696	2,696	100.0%
big_math	27,094	6,728	24.8%
Total	–	40,000	–

To obtain supervision for verification behavior, we prompt GPT-OSS-20B (our target model for training) to verify each draft solution with the following prompt template, producing a verdict and accompanying verification rationale.

Verification Prompt: Training Data Curation

```
In this task you will be given a question and a draft solution to that
question including the thinking steps.  Please walk through the draft
solution sequentially and check if there are any fatal mistakes in
the reasoning (e.g.  incorrect use of theorems / knowledge, incorrect
calculations, etc.).  Please note that once you spot an fatal error
reasoning process, you **MUST** stop checking the remaining steps
and directly output the verdict.  **DO NOT ATTEMPT TO RESOLVE THE
QUESTION**.  If you can find a potential fix to the fatal step, you
can also output it, but you **MUST NOT** attempt to further resolve
the question.  If all steps appears to be correct, you should check the
entire draft.  In the end you must output an overall verdict to the
correctness of the provided draft in the form of:

<overall_verdict>[Correct / Incorrect]</overall_verdict>
<overall_confidence>[High / Medium / Low]</overall_confidence>
<fix>[Any potential fix to the incorrect step if there is any]</fix>

Please be strict with algebra, logic, and theorem use.  Please only
verify and provide feedback to the provided draft.  You **MUST NOT**
attempt to re-solving the problem.

The question provided is:
-- beginning of question --
{problem}
-- end of question --

Please follow the instructions above and examine the draft solution:
-- beginning of draft --
{draft}
-- end of draft --

Please note that once you spot a fatal error reasoning process, you
**MUST** stop checking the remaining steps and directly output the
verdict.  If all steps appears to be correct, you should check the
entire draft.  **DO NOT ATTEMPT TO RESOLVE THE QUESTION**
```

The prompt above is designed such that we can accurately parse the model's final verification verdict wrapped between <overall_verdict> and </overall_verdict>. We compare the model's

verification verdict with the symbolic verifier's result and retain only verification trajectories that agree with the verifier outcome. This produces a set of correct verification trajectories for both correct and incorrect drafts.

We construct the SFT dataset by sampling: (i) $40{,}000$ (question, correct draft, correct verification) triples and (ii) $40{,}000$ (question, incorrect draft, correct verification) triples. This yields $80{,}000$ total training examples with a balanced mixture of correct / incorrect drafts. Each example, depending on whether it contains a correct or incorrect draft, is formatted using a dedicated template (as shown below). `verification_reasoning_trace` is the reasoning trace collected in Section G.2 under the verification prompt, and `clean_slate_reasoning_trace` is the reasoning trace generated by GPT-OSS-20B when correctly solving the problem under the DIRECT setting with no additional context. The (incorrect-draft) trajectory thus realizes an explicit *context reset*: after the verification verdict, the trace switches to a DIRECT-style trajectory that does not reference the draft.

Synthetic Reasoning Trace: Correct Draft

```
<|channel|>analysis<|message|>We need to check the draft.  If it is
correct, I can reuse it as the final answer.  If it is wrong, I need
to come up with a correct solution myself, reusing any useful pieces if
possible.  {verification_reasoning_trace}

We can see that the draft is Correct.  I can reuse the draft solution
as the final answer.
<|end|><|start|>assistant<|channel|>final<|message|><overall_verdict_1>
Correct</overall_verdict_1>

The final answer is:

{final_answer}
```

Synthetic Reasoning Trace: Incorrect Draft (Context Reset)

```
<|channel|>analysis<|message|>We need to check the draft.  If it is
correct, I can reuse it as the final answer.  If it is wrong, I need
to come up with a correct solution myself, reusing any useful pieces if
possible.  {verification_reasoning_trace}

We can see that the draft is Incorrect.  Since the draft is not
correct, I need to create a new solution.  I need to think about this
problem again.

{clean_slate_reasoning_trace}

<|end|><|start|>assistant<|channel|>final<|message|><overall_verdict_1>
Incorrect</overall_verdict_1>

The final answer is:

{final_answer}
```

G.3 TRAINING SETUP

We perform full supervised fine-tuning for one epoch on the $80{,}000$ synthesized trajectories. The objective is a standard *masked cross-entropy loss on the output (assistant-channel) tokens only*: tokens belonging to the prompt (the problem and in-context draft) are masked out and contribute no gradient, so the model is trained exclusively to predict the verification trajectory followed by either the answer-reuse template (correct draft) or the clean-slate continuation that realizes the context reset (incorrect draft). We use cosine learning rate decay with a peak learning rate of 5×10^{-5} and a 10% warmup fraction. Training was conducted on $8\times$H100 nodes with Llama-factory (Zheng et al., 2024).

Table 29: **Targeted SFT generally improves robustness under erroneous drafts**: Accuracy on competitive math benchmarks under 1F, and on the post-hoc subset where the model self-verifies the draft as "Incorrect" (Section 3.2). "Finetuned" denotes GPT-OSS-20B after our conditional fallback training; deltas in parentheses are absolute percentage-point changes vs. baseline. 1F uses the cross-drag setting (drafts from the task's anchor pool): the baseline row matches GPT-OSS-20B in Table 10, not the self-drag numbers in Table 3.

SETTING	AIME24*	AIME25*	HMMT24*	HMMT25*
1F				
Baseline	17.5	18.8	13.3	21.2
Finetuned	40.6 (+23.1)	20.7 (+1.92)	19.2 (+5.89)	27.8 (+6.56)
1F (Conditioned on Self-Detected Error Signal)				
Baseline	28.5	43.3	16.4	33.9
Finetuned	52.9 (+24.4)	38.7 (-4.65)	30.5 (+14.0)	45.3 (+11.4)

G.4 RESULTS

Conditional fallback training improves 1F but does not recover DIRECT. As shown in Table 29, conditional fallback training yields consistent gains in the 1F setting on competitive math benchmarks, indicating that even a lightweight context-reset rule can measurably reduce the degradation induced by erroneous context. However, the fine-tuned model does not fully recover DIRECT performance: although the gap narrows, 1F accuracy remains below DIRECT, suggesting that residual contextual drag persists even after training the context-reset mechanism.

Conditional fallback training improves the model's ability to act on self-detected error signals. We repeat the post-hoc conditioned analysis from Section 3.2, restricting the analysis to cases where the model correctly identifies the draft as incorrect before producing its final answer. Under this evaluation, the fine-tuned model generally improves relative to its pre-tuning counterpart across benchmarks (with the exception of AIME25). This provides a complementary view of training effectiveness: beyond improving unconditional 1F accuracy, conditional fallback training strengthens the model's capacity to translate error awareness into improved downstream reasoning.

Improved robustness comes at the cost of reduced utilization of correct context. We additionally evaluate the trained model on the 1T setting (correct in-context draft) using the same questions and draft-construction pipeline as our main 1T and DIRECT evaluations, with the only exception that the in-context draft is sampled to be correct as determined by the symbolic verifier.

Table 30: **Conditional fallback training decreases 1T performance.** Pass@1 accuracy on competitive math benchmarks under 1T (verify-then-answer) for GPT-OSS-20B baseline and our finetuned checkpoint. Deltas in parentheses are absolute percentage-point changes vs. baseline.

SETTING	AIME24*	AIME25*	HMMT24*	HMMT25*
1T (PASS@1)				
Baseline	73.12	73.56	70.31	76.79
Finetuned	50.63 (-22.49)	49.04 (-24.52)	44.92 (-25.39)	49.11 (-27.68)
1T (PASS@5)				
Baseline	100.00	94.71	95.31	97.02
Finetuned	83.75 (-16.25)	90.87 (-3.84)	85.94 (-9.37)	85.42 (-11.60)

Table 30 shows that while conditional fallback training improves robustness in the 1F setting, it incurs a 22–28 pp absolute drop on 1T pass@1 across all four hard-math benchmarks (and 4–16 pp on pass@5). A plausible explanation is a *robustness-utilization tradeoff*: training the model to context-reset upon detecting errors may also reduce its ability to capitalize on useful information in the provided draft, even when that context is partially or fully correct. In 1T, this can manifest as

overly conservative behavior (discarding potentially helpful structure) or increased token/attention budget spent on verification-style trajectories rather than constructive problem solving. Together, these results indicate that conditional fallback training is not an effective mitigation: on average its 1T cost exceeds its 1F gain, and 1F stays below DIRECT.

G.5 COMPARISON TO OUTCOME-BASED RL (GRPO)

In addition to conditional fallback training, we also experimented with standard Group Relative Policy Optimization (GRPO; Shao et al. 2024) as a post-training alternative, to test whether outcome-based RL alone could induce the context-reset behavior. We train on the *same* set of math problems used for SFT, but optimize only a final-answer correctness reward. Concretely, for each prompt we sample a group of n=8 rollouts from the current policy, compute a binary reward based on whether the final answer matches the ground truth, and apply the GRPO update. We use batchsize 64, maximum context length 16,384, and otherwise follow standard GRPO defaults (e.g., on-policy sampling, per-step KL regularization to the reference policy, and reward normalization within each group). We train for 800 steps, by which point the reward had plateaued, using substantially more compute than the SFT run.

Table 31: **GRPO results on competitive math under contextual drag (1F).** We report accuracy (%) in the 1F setting. "Finetuned" denotes our targeted SFT model; "GRPO" denotes the model trained with standard GRPO. Deltas in parentheses are absolute percentage-point changes relative to the BASELINE in the same block. As in Table 29, the baseline is the cross-drag 1F setting.

SETTING	AIME24*	AIME25*	HMMT24*	HMMT25*
1F				
Baseline	17.5	18.8	13.3	21.2
Finetuned	40.6 (+23.1)	20.7 (+1.9)	19.2 (+5.9)	27.8 (+6.6)
GRPO	22.5 (+5.0)	20.7 (+1.9)	17.6 (+4.3)	24.4 (+3.2)

Table 31 shows that standard GRPO yields modest gains under contextual drag, improving over the baseline by $+1.9$ to $+5.0$ points across benchmarks, and substantially underperforming conditional fallback training. This gap is consistent with GRPO's reward being defined solely on final-answer correctness: while it can nudge the policy toward better outcomes on average, it does not explicitly teach the model to context-reset its reasoning trajectory after detecting errors in the in-context draft. In contrast, the supervised objective directly trains the post-verification trajectory to match the DIRECT reasoning trace, which appears more aligned with mitigating contextual drag. Overall, these results suggest that naïvely applying outcome-based RL is not a drop-in replacement for targeted supervision in this setting, and that more structured rewards (e.g., verifier-aware or trajectory-level signals) may be needed for RL-style training to reliably counteract erroneous context.