DR.RL-BENCH: DIAGNOSING REINFORCEMENT LEARNING RUNS WITH AI AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Autonomous reinforcement-learning (RL) post-training requires agents to understand a training run before deciding how to improve it. Existing benchmarks primarily measure end-to-end model improvement, conflating training diagnosis with intervention design and experiment execution. We introduce Dr.RL-Bench, a benchmark that isolates diagnosis as an evidence-grounded reasoning task before intervention. Its 196 tasks are constructed from real GPU training runs and cover anomalies in data, training parameters, infrastructure, rewards, and environments. Agents freely investigate the provided code, configurations, logs, and generated responses, then report checkable Facts and evidence-supported Inferences. Expert-authored rubrics evaluate whether these findings identify relevant issues and explain their underlying causes. Experiments with eight LLM agents reveal substantial diagnostic gaps: the strongest model achieves 47.52/100, and all agents score lower on Inferences than on Facts. These results and case studies show how agents can recognize symptoms yet fail to connect dispersed evidence into a supported explanation. Dr.RL-Bench complements outcome-based evaluation by assessing what agents understand about a training run before they act.

1 INTRODUCTION

Reinforcement learning (RL) post-training is increasingly used to improve LLM agents that interact with tools and environments over multiple steps. Building an effective training pipeline requires choices about data, rewards, optimization, and runtime systems, as well as repeated inspection of training signals and agent behaviors. This motivates autonomous RL post-training, where an LLM agent analyzes a training run, decides what to change, executes another experiment, and updates its strategy based on the observed results. The goal is to continuously improve agent performance under a limited computational budget.

Existing benchmarks for autonomous post-training, such as PostTrainBench (Rank et al., 2026) and Agent^2 RL-Bench (Chen et al., 2026), evaluate whether an agent can improve an LLM through an end-to-end training process (Rank et al., 2026). However, the final performance improvement entangles multiple capabilities, including training diagnosis, intervention design, and experiment execution, making it difficult to identify which capability contributes to the success or failure (as shown in Figure 1 (a)).

To better isolate individual capabilities, recent benchmarks evaluate specific aspects of autonomous model development, including data curation in Curation-Bench (Kang et al., 2026), data-centric research in RSIBench-Data (Meng et al., 2026), and fine-tuning efficiency in RE-Bench (Wijk et al., 2024). Nevertheless, these benchmarks mainly focus on whether an agent can perform downstream optimization tasks or achieve improved outcomes, while the ability to diagnose an existing training run before deciding what to change remains unexplored.

This diagnostic capability is critical for autonomous RL post-training because diagnosis serves as the first step of the iterative optimization loop. Before selecting any intervention, an agent must first infer the underlying causes behind the observed training outcomes. However, inaccurate diagnosis can negatively affect both the effectiveness and efficiency of the optimization process. On the effectiveness side, incorrect interpretations may lead the agent to apply inappropriate interventions, repeatedly optimize toward wrong assumptions, and accumulate biased feedback across iterations.

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.

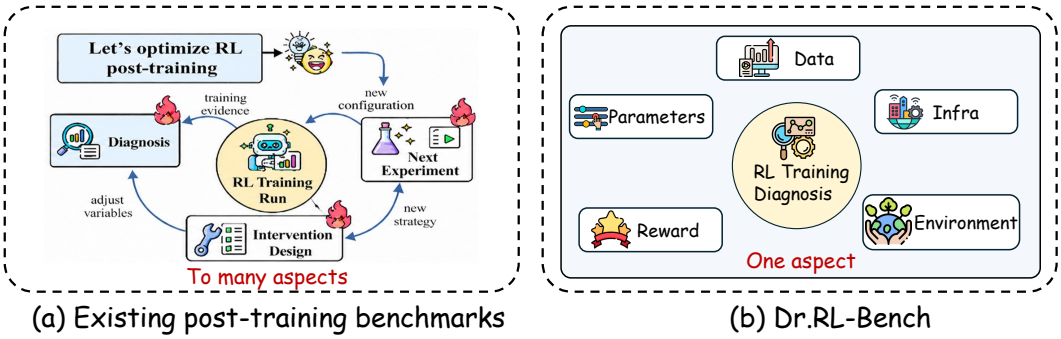


Figure 1: A comparison of existing benchmark.

On the efficiency side, such failures can introduce unnecessary trial-and-error experiments, increasing computational costs and delaying convergence. Therefore, reliable diagnosis is essential for enabling effective and efficient autonomous RL post-training.

To bridge this gap, we introduce Dr.RL-Bench, a benchmark designed to evaluate the diagnostic capability of LLM agents in autonomous RL post-training. Unlike existing benchmarks that directly evaluate whether an agent can improve final model performance, Dr.RL-Bench isolates the diagnosis stage as the first step of the autonomous optimization pipeline. It provides completed RL training runs and evaluates whether agents can identify potential issues, infer their underlying causes, and generate evidence-grounded explanations before taking further actions.

Specifically, as shown in Figure 1 (b), Dr.RL-Bench constructs diverse RL training scenarios covering representative failure modes across the RL post-training pipeline, including data quality, training parameters, infrastructure configurations, reward design, and environment settings. For each scenario, agents are provided with available training evidence, such as learning curves, evaluation results, configuration information, and behavioral observations. The benchmark then evaluates whether generated diagnoses are consistent with the underlying causes, enabling a controlled assessment of an agent's ability to interpret RL training processes before intervention.

Our contributions are summarized as follows:

- We introduce **Dr.RL-Bench**, a benchmark that isolates the diagnosis stage in autonomous RL post-training and evaluates LLM agents' ability to identify underlying training issues before intervention.
- We formulate RL training diagnosis as an evidence-grounded reasoning task and construct diverse scenarios covering representative failure modes in autonomous RL pipelines.
- We conduct extensive evaluations of representative LLM agents and analyze their capabilities and limitations in diagnosing RL training processes.

2 Related Work

2.1 Benchmarks for Autonomous LLM Post-Training

Recent benchmarks study whether LLM agents can autonomously improve models through post-training. Existing efforts can be broadly divided into two categories: end-to-end autonomous post-training benchmarks and component-level capability benchmarks.

End-to-End Autonomous Post-Training. PostTrainBench (Rank et al., 2026) and Agent^2 RL-Bench (Chen et al., 2026) evaluate whether agents can complete an end-to-end post-training pipeline and improve model performance through iterative experimentation. However, their final outcomes entangle multiple capabilities, including training diagnosis, intervention design, and experiment execution, making it difficult to isolate the contribution of each capability.

Component-Level Capability Evaluation. Recent benchmarks further investigate specific stages of autonomous model development. Curation-Bench (Kang et al., 2026) evaluates autonomous data curation, RSIBench-Data (Meng et al., 2026) studies data-centric research capabilities, and RE-Bench (Wijk et al., 2024) evaluates AI research engineering abilities. Nevertheless, these benchmarks mainly focus on downstream optimization or execution capabilities, while the upstream capability of diagnosing an existing training run before intervention remains unexplored. In this paper, we address this gap by introducing a benchmark that isolates RL training diagnosis as the first stage of autonomous RL post-training and evaluates agents' ability to identify underlying issues before designing subsequent interventions.

3 Benchmark Design

Each Dr.RL-Bench task asks an agent to understand and assess an RL post-training run through a task package containing its code, configurations, logs, and generated responses. The agent can freely explore the provided training scene, e.g., by inspecting code, analyzing logs, or writing and running analysis scripts. It reports findings it considers noteworthy as *Facts* (checkable observations) and *Inferences* (conclusions supported by those observations). For example, an agent might identify that validation samples were included in the training data (Fact) and infer that the evaluation results may therefore be inflated by data contamination (Inference). See Appendix C (Tables 5, 6, and 8) for more detailed examples.

Figure 2 outlines our benchmark construction pipeline. Experts (PhD-student level or above, with RL research and engineering experience) review research papers, public web resources, and internal engineering documents to organize target anomalies across the RL post-training pipeline and gather their candidate interventions. These interventions are then implemented and tested in real GPU runs with controlled ablations to verify the hypothesized causes. Artifacts from validated runs are filtered and assembled into task packages. Independent expert annotation and consensus then establish which Facts and Inferences the package supports and whether its materials require revision. Accepted package revisions return to the packaging stage and restart all annotation from scratch. Each finalized task pairs an agent-facing package with a hidden rubric.

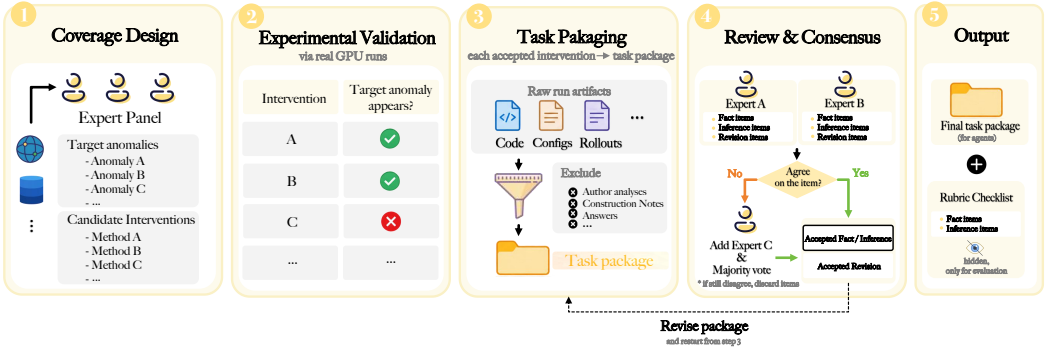


Figure 2: Dr.RL-Bench construction pipeline. See Appendix A.3 for coverage-design (Table 4).

3.1 Anomaly Curation and Experimental Validation

We organize anomalies into five broad problem domains. Examples include issues with training-data selection, sampling, and consumption (*Data*); optimization settings and their effects on updates (*Parameters*); training and inference implementations, numerical computation, and runtime behavior (*Infra*); reward computation, attribution, and alignment with intended behavior (*Reward*); and interactions, state transitions, and termination (*Environment*). These examples are not exhaustive; further illustrations and task distributions appear in Appendix B.

We test the candidate interventions in actual GPU training runs, retaining those that reproduce the target behavior. Controlled ablations then compare otherwise matched conditions with and without the proposed cause, checking that the anomaly appears when the cause is introduced and is absent

when it is removed. These experiments validate the cause under the tested conditions. Appendix A.3 illustrates the mapping from target anomalies to candidate interventions across the five domains.

3.2 Task packaging and neutral handoffs

Each task presents a *training scene* frozen at an observation time t: the code, configurations, logs, metrics, and trajectories available from the beginning of the run through that point. We assemble this scene into a complete task package, filtering out construction-only auxiliary files, author analyses, intervention descriptions, and answer hints. Validation controls and ablations remain author-side. We retain artifacts naturally available during training, even when they directly reveal an anomaly. This avoids artificially increasing difficulty by withholding routine training materials and ensures that the evidence needed to solve the task remains available. The agent receives a neutral handoff asking it to inspect the materials and report all findings it considers noteworthy as Fact and Inference items, without the target anomaly description, rubric, or a prescribed investigation path. Appendix A.1 shows an example of the task card received by the agent. Further details on task packaging and artifact filtering are provided in Appendix A.

3.3 Expert annotation, consensus, and revision

Two experts independently review each package with knowledge of the target anomaly and propose Fact, Inference, and package-revision items. Agreed items are accepted; disagreements are discussed with a third expert and resolved by majority support. Accepted revisions trigger repackaging and a full restart of annotation; reconciliation, rejection, and revision rules are detailed in Appendix A.4.

3.4 Evaluation protocol

We assess whether an agent identifies the anomalies of interest in the training scene and reports the key Facts and Inferences needed to understand them. Following the LLM-as-a-Checker paradigm (Liu et al., 2026), we use GPT-5.6 Luna (OpenAI, 2026) at max reasoning effort to check the reported list against individual hidden rubric items, accepting semantically equivalent findings. For each task, Fact and Inference coverage are the fractions of rubric items matched within each group, with equal item weights. The task score is 30 times Fact coverage plus 70 times Inference coverage, and task scores are averaged equally across the benchmark.

This 3:7 weighting is used because we found that, in such an open-ended, self-directed investigation setting of Dr.RL-Bench, Facts can be scored more easily by restating code fragments or log entries (even when agents don't uncover the target anomaly), whereas Inferences typically require connecting dispersed evidence through sound reasoning, a capability more central to anomaly diagnosis. As shown in the case studies (Sections 4.2.1 and 4.2.2), agents can detect repetitive generation or discrepancies between training and inference probabilities yet fail to understand the underlying inference items.

4 Experimental Results

4.1 Evaluation setup and overall results

We evaluate eight agents on the same 196 tasks with a common harness built on DeepAgents (LangChain, 2026), providing filesystem access and CPU analysis tools but no GPU access. The models are Claude Opus 5 (Anthropic, 2026), Qwen3.8-Max-0902 (Alibaba Cloud, 2026b) and Qwen3.7-Max (Alibaba Cloud, 2026a), GLM-5.3 (Z.ai, 2026b) and GLM-5.2 (Z.ai, 2026a), Kimi K3 (Moonshot AI, 2026), and DeepSeek V4.1 Flash (DeepSeek-AI, 2026a) and V4 Pro-0813 (DeepSeek-AI, 2026b).

Experimental results show that RL anomaly diagnosis remains challenging for current agents (Table 1). Even Opus 5, the highest-scoring model overall and across all five domains, earns less than half of the available diagnostic credit. We observe a consistent and interesting pattern: all agents score lower on Inference than on Fact items. Qwen 3.8 Max, the second-best model overall, achieves nearly the same Fact score as Opus 5 but falls further behind on Inference items. Figure 4 shows

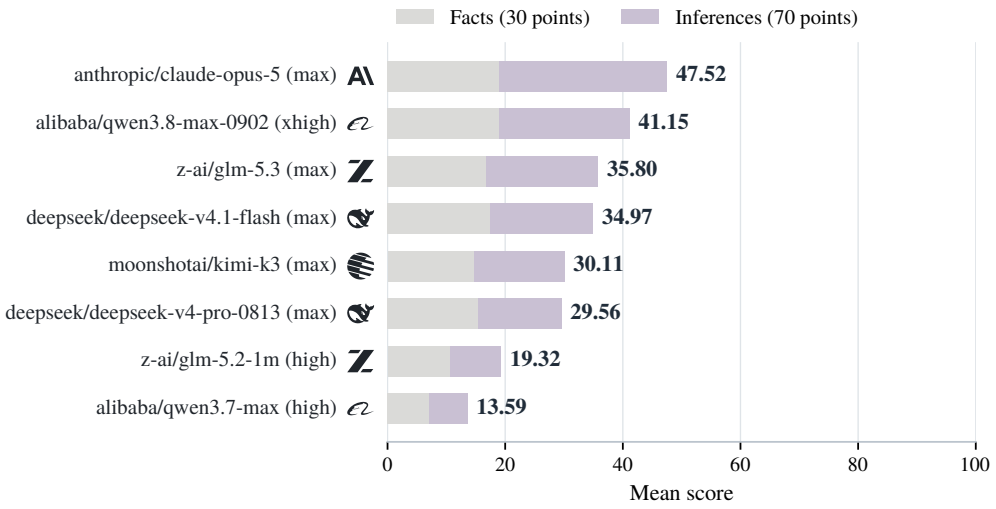


Figure 3: Diagnostic coverage on the same 196 tasks for all eight agents. Bars show equal-task mean scores; segments are the Fact and Inference contributions, capped at 30 and 70 points respectively. Parentheses indicate provider-specific reasoning settings.

Table 1: Main results. All metrics cover the same 196 tasks. Domain headers give task counts; bold marks the highest score. Hit ratio (%) is the pooled ratio of rubric hits to submitted items. Efficiency reports mean completed model calls (turns) and mean Answerer wall time per task, excluding Judge time.

	Diagnostic score			Scores by domain					Report	Efficiency	
Model	Overall	Fact	Inf.	Data (36)	Param. (51)	Infra (54)	Reward (34)	Env. (21)	Hit ratio (%)	Turns	Time (min)
Opus 5	**47.52**	**63.64**	**40.61**	**46.86**	**50.92**	**44.08**	**49.24**	**46.47**	30.12	41.28	57.60
Qwen 3.8 Max	41.15	63.40	31.61	36.02	41.92	40.84	45.35	42.04	29.63	44.22	31.78
GLM-5.3	35.80	55.99	27.15	33.85	37.86	32.48	40.25	35.52	34.95	45.78	17.71
DeepSeek V4.1 Flash	34.97	58.46	24.90	35.53	34.56	31.81	38.44	37.54	30.00	47.61	30.30
Kimi K3	30.11	49.27	21.89	29.08	30.16	28.15	34.21	30.13	34.31	31.50	22.34
DeepSeek V4 Pro	29.56	51.06	20.35	30.90	28.48	27.42	35.36	26.00	30.42	40.72	20.87
GLM-5.2	19.32	35.67	12.32	17.42	20.37	16.41	24.73	18.79	24.76	44.67	7.92
Qwen 3.7 Max	13.59	23.66	9.27	12.74	14.93	11.28	17.47	11.43	19.20	25.83	14.62

that more elapsed time does not consistently imply a higher score: GLM-5.3 scores above DeepSeek V4.1 Flash while using less observed time, whereas Opus leads in score with the longest time.

In the next section, we'll examine three detailed case studies to illustrate what different agents discover, which key inferences they miss, and how these findings are reflected.

4.2 Case studies

The following cases examine stopping degradation, training–inference disagreement, and redundant input preparation. Agents receive raw training scenes, not the explanatory figures or diagnostic summaries shown here. We distinguish their autonomous diagnoses from separate *rubric-guided repairs*, in which the cause is disclosed before implementation; these repairs do not contribute to diagnostic scores. Detailed evidence, complete rubrics, and extended repair comparisons appear in Appendix C. The three cases require different connections: relating a loss's support to stopping behavior, relating a numerical discrepancy to attention implementation, and relating runtime cost to input-construction code. They illustrate how partial findings can fall short of a diagnosis.

4.2.1 STOPPING DEGRADATION IN ON-POLICY DISTILLATION

This case distills a Qwen3-8B mathematics teacher into a Qwen3-1.7B-Base student on DAPO-Math-17k (Yu et al., 2025). Following the on-policy distillation paradigm (Agarwal et al., 2024), the student generates answers, and the teacher supplies next-token distributions along those sequences. Some initially terminating answers develop excessive continuation after training: one validation answer shows verbose repeating and grows from 534 tokens to the 2,048-token limit.

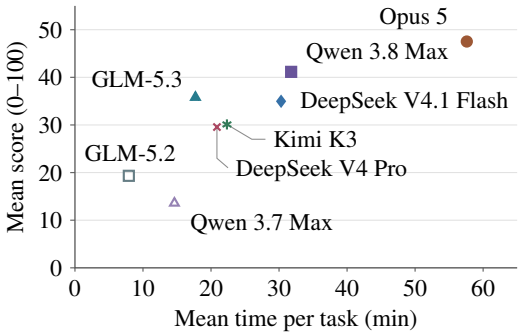


Figure 4: Overall score versus mean Answerer wall time, excluding Judge time.

At answer-ending positions, the student favors the end-of-sequence (EOS) token `<|endoftext|>`, whereas the teacher favors `<|im_end|>`. This difference is not a decoding incompatibility: both token IDs are already accepted as stops. An accepted stop token is a condition on generation, not a guarantee that the model assigns it substantial probability. Since the student would stop on either identity, the question is how training changes its tendency to produce one of them. The mechanism instead involves the distillation objective, which selects the student's top-16 tokens, retrieves teacher probabilities on the same IDs, and separately renormalizes both distributions before reverse KL. Writing S for the student-selected set and $\widetilde{p}_S$ and $\widetilde{q}_S$ for the student and teacher distributions normalized only over S, the per-position objective is

$$\mathcal{L}_S = \sum_{v \in S} \widetilde{p}_S(v) \log \frac{\widetilde{p}_S(v)}{\widetilde{q}_S(v)}.$$

The separate normalization matters: teacher logits outside S cannot change this target distribution, even when an excluded token is highly probable under the teacher's full vocabulary distribution. At the probed ending position (Figure 5), the student's EOS is included but receives little teacher probability, so the loss suppresses its logit. The teacher's preferred EOS is excluded and receives no direct supervision from that position's loss. Together, these effects provide a mechanism by which the objective can weaken existing stopping behavior without teaching the alternative.

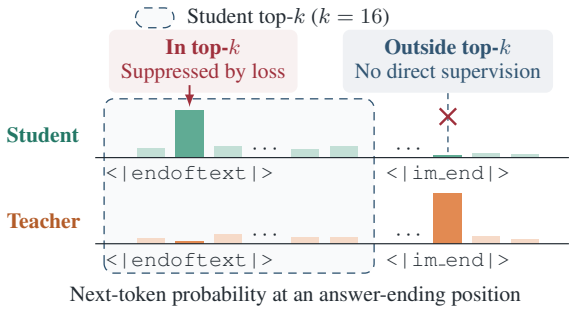


Figure 5: EOS supervision at one probed ending position. The dashed region spans the student-selected top-16 IDs in both distributions. Heights are schematic; suppression and absent supervision refer to local logit gradients.

Evidence available to the agent.
Training and validation rollouts show the change in answer endings, while generation configurations and call arguments establish which tokens actually terminate decoding. Loss source code exposes candidate selection and separate normalization. Available student and teacher weights, tokenizers, and input sequences permit CPU forward probes on the same prefixes. These materials support a comparison of the two stopping preferences, but do not supply a precomputed distribution or gradient analysis. A complete diagnosis must connect the behavior in the rollouts to the tokens included in, and excluded from, the objective at relevant ending positions.

Agent diagnosis. Opus 5 and Kimi-K3 tie at 38/100. Both identify repetitive continuation, acceptance of both EOS tokens, and the student's observed use of `<|endoftext|>`. They also

recognize that tokens outside the student-selected set receive no direct supervision. Opus tests this property: adding 10,000 to every excluded teacher logit leaves the loss unchanged. Kimi treats the truncation as a reasonable approximation. Neither checks the teacher's preference at the corresponding ending positions or asks whether an excluded token is essential to stopping. The gap is therefore not simply failure to read the code, but failure to connect a demonstrated supervision blind spot to the observed behavior. Calling top-k truncation an approximation does not settle whether it is adequate for this run: the missing question is which excluded tokens matter for the behavior under investigation. Similarly, checking that both EOS IDs are accepted rules out a decoding-configuration explanation, but does not test the training signal that makes either EOS likely to be generated.

Rubric-guided repair. The agent swaps the teacher's target logits for the two EOS identities before candidate normalization, redirecting its stopping signal to the student's existing EOS. For the validation example above, the answer after the second update changes from a 2,048-token continuation to a normally terminated 514-token answer. Appendix C.2 includes the full criteria and additional before/after outputs. The comparison uses the same initial weights, validation questions, and sampling settings. A second example also stops after repair instead of repeating numbers to the limit. These results concern restoring termination.

4.2.2 LSE MISMATCH IN INFERENCE ATTENTION

We train a 4B Qwen3-architecture model with GRPO on GSM8K (Cobbe et al., 2021) mathematics problems. Inference records each generated token's probability; the learner recomputes probabilities for the same answer before updating the policy. With matching weights, temperature, and input prefix, these estimates should agree within numerical error. Instead, about 7.5% of valid tokens have log-probability differences greater than 1, and the maximum approaches 13. An absolute difference greater than 1 means that the probabilities differ by a factor greater than $e \approx 2.7$, making the tail important even when the median discrepancy is small. Some recorded answers also contain repetitive digits, but the probability disagreement itself supplies the direct computational anomaly.

The defect lies in inference-side attention. Cascade Attention (Ye et al., 2024) computes local outputs for separate prefix and suffix blocks, then merges them with weights determined by each block's log-sum-exp (LSE). This is equivalent to full attention only when each query head's local outputs are paired with its own LSE values (Figure 6). For local outputs $\mathbf{O}_P, \mathbf{O}_S$ and corresponding LSE values L_P, L_S, the merge is

$$\mathbf{O} = \alpha \mathbf{O}_P + (1-\alpha)\mathbf{O}_S, \qquad \alpha = \frac{e^{L_P}}{e^{L_P} + e^{L_S}}.$$

For scores s_t in block A, $L_A = \log \sum_{t \in A} e^{s_t}$ records the block's normalization mass. It determines how much that block contributes to the joint softmax, rather than treating the two locally normalized outputs as equally weighted.

In the executed FlashAttention 2 (Dao, 2024) path, a return-interface transpose misinterprets the layout produced with Grouped-Query Attention (Ainslie et al., 2023). Multiple query heads share one key/value head, and the kernel's output groups LSE values accordingly. It writes the values in key/value-head, group, sample order, where the group dimension distinguishes query heads sharing a key/value head. The return interface instead transposes them as though their order were key/value-head, sample, group. This is a mismatch between the producer's layout and the consumer's interpretation, not an inaccuracy in computing the individual LSE values. The tensor shape remains valid, but some LSE values become associated with the wrong query and attention

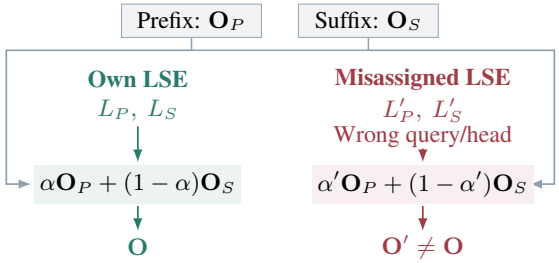


Figure 6: LSE mismatch, schematically. The local output vectors stay fixed; misassigned LSE values change their merge weights and can change the output. Primes denote the mismatched computation.

head. The resulting merge can use incorrect weights even when its local output vectors are correct, changing the final attention output and subsequent token probabilities. Appendix C.3 gives the layout and merge derivation, with the full attention comparison in Figure 11.

Evidence available to the agent. Inference logs and saved training batches store the two probability records separately. Agents must interpret the producing code, match the same question and generated token sequence, and compare corresponding positions; a plot of disagreement is not supplied. Configuration and synchronization code establish the conditions for that comparison. Locating its cause additionally requires following the attention backend from LSE storage through its return interface to Cascade Attention's merge. A transpose is not intrinsically wrong: the relevant test is whether it preserves the pairing between each query head's LSE and local output, not merely the expected tensor shape.

Agent diagnosis. Opus 5 scores highest at 35.33/100. All agents notice repetitive digits; Opus, Qwen3.8-Max, and GLM-5.3 additionally join the separately stored training and inference records and recover large probability differences. Opus recognizes the mismatch but suggests that a flat predictive distribution could amplify small logit deviations. Qwen3.8-Max emphasizes the small median difference, while GLM ultimately attributes the disagreement to normal numerical error despite initially recognizing that the large differences warrant investigation. None traces the discrepancy to LSE layout and output merging. These reports show that finding an anomaly does not ensure that an agent tests whether its eventual explanation accounts for it. In particular, a small median cannot account for the large tail that these agents already reconstructed. The unresolved step is to test a proposed numerical explanation against those observations and trace where the two forward computations diverge under matching conditions.

Rubric-guided repair. Given the rubric checklist, Opus 5 replaces the erroneous transpose with the appropriate reshape, restoring LSE/output correspondence. The fraction of tokens with absolute training–inference log-probability differences greater than 1 falls from 7.5% to zero, while the mean falls from 0.326 to 0.010. Two validation examples also change from digit continuations at the generation limit to correct, normally terminated calculations (Appendix C.3). These output comparisons use the same questions and decoding settings during initial validation, before optimizer updates. They connect the implementation correction to actual generation behavior, beyond the intermediate attention mismatch.

4.2.3 Redundant tokenization in multi-turn agentic training

We train Qwen3-8B (Yang et al., 2025) with GRPO (Shao et al., 2024) on ALFWorld (Shridhar et al., 2021) to follow household instructions, such as placing objects in a refrigerator. Each action produces environment feedback that the model uses to choose its next action. Preparing the next model input becomes slower as the interaction history grows, although the newly appended content stays similar in size. This trend concerns token construction after feedback arrives, not environment execution or model generation (Appendix C.1).

The verl implementation (Sheng et al., 2024) used in this case assembles the next input by concatenating saved history token IDs, new feedback IDs, and an assistant generation prefix that marks the start of the next response. Only the latter two components need to be newly obtained. Feedback includes the observation and available actions. To encode it, the implementation applies the chat template to a short auxiliary message list, then to that list with the new feedback appended; removing the common prefix leaves the feedback IDs. Neither encoding needs the full interaction history.

Figure 7 contrasts the two extraction paths. Prefix extraction uses the same suffix-difference idea, but applies the template to the complete history twice, without and with the generation prefix. It retains only the extra suffix, which is the same short sequence throughout this run. Thus, even though the final input reuses saved history IDs, obtaining the prefix recomputes and discards them at every turn. The work grows with history length. The defect is the context scope of prefix extraction, not the use of two template calls or suffix differencing in itself. The prefix's constancy in this run does not imply that a reusable framework should hard-code it: its form can vary with the model and template settings. The relevant contrast is that the implementation already obtains one new component without re-encoding the history, but does not apply that strategy to the other.

Evidence available to the agent. Raw per-turn timing records and their producing code establish that the measured interval covers token construction and merging after feedback arrives, rather than the environment's response time. Trajectories expose the growing inputs, feedback messages, and appended IDs; the tokenizer and template configuration make the short prefix identifiable. Connecting these records to the two extraction paths is essential: rising latency alone could look like an unavoidable consequence of longer context, while saved-ID reuse alone could make input construction appear already incremental.

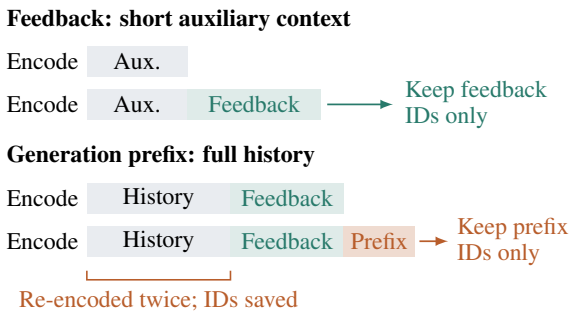


Figure 7: Two encodings per branch retain only a suffix. Green denotes feedback. Prefix extraction repeatedly re-encodes history whose IDs are already saved. Widths are schematic.

Agent diagnosis. Opus 5 scores highest at 18/100, identifying rising preparation time, saved history-ID reuse, and the constant short prefix. The other evaluated agents score zero on this case's rubric. Yet even Opus does not connect these observations to the scope of encoding inside prefix extraction. Several reports investigate other issues, such as zero advantages within GRPO groups, without explaining the preparation overhead. The missing step is to connect the timing trend to the actual work performed by the input-construction code. Those other findings need not be wrong, but leave this particular cost-exploding issue unexplained. Recognizing that the prefix is constant is insufficient without checking how its IDs are obtained on each turn.

Rubric-guided repair. After getting the rubric, Opus successfully changes prefix extraction to use the short auxiliary-context difference already used for feedback. In same-input CPU replay, median preparation time rises from 3.24 to 27.49 ms across measured turns before repair but stays at approximately 0.91–1.00 ms afterward (Figure 10 in Appendix C.1). This removes the pronounced history-dependent cost in the measured operation. Replaying identical recorded inputs separates this change from variations in trajectory length or environment feedback. The repair applies the existing incremental strategy consistently to both new input components; it does not require treating longer interaction histories as a fault or discarding information from the model's context.

Cross-case implications. The cases reveal different stopping points in autonomous diagnosis: a verified supervision blind spot not linked to stopping behavior, a reconstructed probability mismatch left without an implementation-level explanation, and unconnected facts about redundant computation. Rubric-guided repairs improve the corresponding behavior once the cause is supplied. Note that this contrast separates implementing a known correction from independently developing an evidence-grounded diagnosis. Dr.RL-Bench focuses on the latter: an agent's ability to autonomously investigate a training scene and develop an evidence-grounded diagnosis.

5 Conclusion

We introduced Dr.RL-Bench to evaluate evidence-grounded diagnosis as a distinct stage of autonomous RL post-training, before intervention design and experiment execution. Its 196 tasks span five problem domains, with Fact and Inference rubrics distinguishing checkable observations from explanations of their relationships and underlying mechanisms.

Evaluation of eight LLM agents reveals a persistent gap between these abilities: agents more readily recover facts than develop the inferences needed to explain anomalies. The case studies illustrate how recognizing symptoms can fall short of connecting dispersed evidence into a supported diagnosis. These gaps make evidence integration and hypothesis testing concrete priorities for agent development. Item-level evaluation can help track progress beyond aggregate scores. Dr.RL-Bench complements outcome-based benchmarks by evaluating what agents understand about a training run before they decide how to intervene.

Ethics Statement

Dr.RL-Bench is intended for research on the reliability of AI agents in RL post-training. Benchmark scores measure diagnostic performance on the provided scenes, not the safety of autonomous deployment. Responsible sharing and reuse of training artifacts requires respecting the licenses of the underlying models and datasets and protecting sensitive information.

Reproducibility Statement

Section 3 describes benchmark construction and the Fact/Inference evaluation protocol. Section 4 reports the evaluated models, reasoning settings, and results. Appendix A provides the task-card example and annotation details, while Appendix C documents supplementary measurements, diagnostic rubrics, and repair comparisons.

AI Use Statement

Generative AI tools are used for translation, language polishing, and grammar correction. The authors retain responsibility for the final manuscript, results, and associated artifacts.

References

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2306.13649.

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. GQA: Training generalized multi-query transformer models from multi-head checkpoints. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023. URL https://arxiv.org/abs/2305.13245.

Alibaba Cloud. Qwen3.7-Max model information, 2026a. URL https://www.alibabacloud.com/help/en/model-studio/qwen3-7-max. Evaluated snapshot: qwen3.7-max-2026-06-08.

Alibaba Cloud. Alibaba unveils Qwen3.8-Max: Its largest and most capable flagship model to date, August 2026b. URL https://www.alibabacloud.com/en/press-room/alibaba-unveils-qwen3-8-max.

Anthropic. Introducing Claude Opus 5, July 2026. URL https://www.anthropic.com/news/claude-opus-5.

Wanyi Chen, Xiao Yang, Xu Yang, Tianming Sha, Qizheng Li, Zhuo Wang, Bowen Xian, Fang Kong, Weiqing Liu, and Jiang Bian. Agentˆ 2 rl-bench: Can llm agents engineer agentic rl post-training? *arXiv preprint arXiv:2604.10547*, 2026.

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

Tri Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2307.08691.

DeepSeek-AI. DeepSeek-V4.1-Flash: Pushing the limits of KV cache compression, September 2026a. URL https://huggingface.co/deepseek-ai/DeepSeek-V4.1-Flash.

DeepSeek-AI. DeepSeek-V4-Pro GA release, August 2026b. URL https://api-docs.deepseek.com/news/news260813/.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://arxiv.org/abs/2106.09685.

Feiyang Kang, Hanze Li, Adam Nguyen, Mahavir Dabas, Jiaqi W Ma, Frederic Sala, Dawn Song, and Ruoxi Jia. Can generalist agents automate data curation? *arXiv preprint arXiv:2606.04261*, 2026.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015. URL https://arxiv.org/abs/1412.6980.

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 29th Symposium on Operating Systems Principles*, 2023. URL https://arxiv.org/abs/2309.06180.

LangChain. DeepAgents: The batteries-included agent harness. Software, 2026. URL https://github.com/langchain-ai/deepagents.

Wei Liu, Peijie Yu, Michele Orini, Yali Du, and Yulan He. Hunt instead of wait: Evaluating Deep Data Research on large language models. *arXiv preprint arXiv:2602.02039*, 2026. URL https://arxiv.org/abs/2602.02039.

Fanqing Meng, Lingxiao Du, Qiguang Chen, Ziqi Zhao, Haocheng Lu, Mengkang Hu, and Michael Qizhe Shieh. Rsibench-data: Benchmarking data-centric research for recursive self-improvement. *arXiv preprint arXiv:2607.25886*, 2026.

Moonshot AI. Kimi K3: Open frontier intelligence, July 2026. URL https://github.com/MoonshotAI/Kimi-K3. Model release and technical report.

OpenAI. GPT-5.6 Luna. Model documentation, 2026. URL https://developers.openai.com/api/docs/models/gpt-5.6-luna.

Ben Rank, Hardik Bhatnagar, Ameya Prabhu, Shira Eisenberg, Karina Nguyen, Matthias Bethge, and Maksym Andriushchenko. PostTrainBench: Can LLM agents automate LLM post-training? *arXiv preprint arXiv:2603.08640*, 2026. URL https://arxiv.org/abs/2603.08640.

Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of Adam and beyond. In *International Conference on Learning Representations*, 2018. URL https://arxiv.org/abs/1904.09237.

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

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

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. HybridFlow: A flexible and efficient RLHF framework. *arXiv preprint arXiv:2409.19256*, 2024. URL https://arxiv.org/abs/2409.19256.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. ALFWorld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations*, 2021. URL https://arxiv.org/abs/2010.03768.

Hjalmar Wijk, Tao Lin, Joel Becker, Sami Jawhar, Neev Parikh, Thomas Broadley, Lawrence Chan, Michael Chen, Josh Clymer, Jai Dhyani, et al. Re-bench: Evaluating frontier ai r&d capabilities of language model agents against human experts. *arXiv preprint arXiv:2411.15114*, 2024.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. URL https://arxiv.org/abs/2505.09388.

Zihao Ye, Ruihang Lai, Bo-Ru Lu, Chien-Yu Lin, Size Zheng, Lequn Chen, Tianqi Chen, and Luis Ceze. Cascade inference: Memory bandwidth efficient shared prefix batch decoding, February 2024. URL https://flashinfer.ai/2024/02/02/cascade-inference.html.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, et al. DAPO: An open-source LLM reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025. URL https://arxiv.org/abs/2503.14476.

Z.ai. GLM-5.2: Built for long-horizon tasks, June 2026a. URL https://z.ai/blog/glm-5.2.

Z.ai. GLM-5.3: Frontier coding with emergent cyber capabilities, August 2026b. URL https://z.ai/blog/glm-5.3.

A Construction and Annotation Details

A.1 Agent-facing task card

Table 2 shows the task card for the on-policy distillation case in Section 4.2.1. It identifies the training method and the scope of the available materials, then asks for an evidence-backed investigation without naming the anomaly or suggesting where to look. Accompanying documentation describes the available materials, analysis tools, and structured report format.

Task

You are taking over an in-progress RL post-training run.

Training method: On-policy distillation for mathematical problem solving.

Data and monitoring scope: An inherited RL post-training handoff with chronological run records, resolved runtime state, learner/update telemetry, and executed source where available.

Investigate the health of this run. You may inspect all files in the task package and freely write and run code for analysis.

The run may be healthy or may contain one or more problems. When finished, submit a structured, evidence-backed report covering all findings you consider worth noting, including the key Facts and Inferences for each finding.

Your investigation report will be independently evaluated and scored by domain experts.

Table 2: Task card example. The target anomaly and diagnostic rubric are not disclosed to the agent.

A.2 Problem domains

Table 3 gives examples of investigation topics within the five problem domains. These domains describe where a problem arises; algorithms, base policies, datasets, and environments describe the settings in which it is instantiated.

Table 3: Problem domains and illustrative investigation topics (non-exhaustive).

Domain	Illustrative investigation topics
Data	Sample selection and exposure; data lineage; links between consumed samples and model behavior.
Parameters	Configured versus effective updates; optimization dynamics; training-state continuity.
Infra	Training and inference implementations; numerical computation; runtime behavior and computational cost.
Reward	Reward computation and attribution; alignment between reward improvement and intended behavior.
Environment	Observations and actions; episode-state transitions; termination and its effects on training trajectories.

A.3 Target anomalies and candidate interventions

Table 4 expands the coverage-design stage in Figure 2 with representative anomalies and candidate induction mechanisms from the 196-task study. These routes provide mechanistic hypotheses for experimental validation, not task-specific reproduction recipes; their effects depend on the model and operating conditions.

Table 4: Representative target anomalies and candidate intervention routes. Rows group related phenomena from the study; they are illustrative examples, not an exhaustive taxonomy or a count of distinct scenarios.

Domain	Target anomaly	Illustrative candidate interventions
Data	Skewed or insufficient sample exposure	Drop configured source weights; count a token-based rehearsal quota in examples; deduplicate independently per worker under a shared update budget.
Data	Train–validation leakage	Split individual turns after flattening conversations instead of preserving conversation-level train/validation membership.
Data	Misbound training content	Reassign images among questions; interpret teacher token IDs from separate vocabulary shards without restoring their global offsets.
Data	Premature trajectory censoring	Count environment-feedback tokens against the model-response budget, causing otherwise usable interaction trajectories to exhaust their allowance.
Parameters	Excessive update magnitude	Increase the effective actor learning rate; reuse the same rollout batch for extra PPO (Schulman et al., 2017) epochs; relax the effective gradient-clipping threshold.
Parameters	Mis-scaled or biased advantages	Add an unintended group-standard-deviation normalization; replace the leave-one-out mean reward baseline with the minimum peer reward.
Parameters	Trainable adapters fail to learn	Omit attached LoRA (Hu et al., 2022) parameters from optimizer groups; initialize both low-rank factors to zero, eliminating their initial learning signal.
Parameters	Optimizer discontinuity on resume	Omit AMSGrad's (Reddi et al., 2018) historical maximum second moment; change effective Adam (Kingma & Ba, 2015) momentum or weight-decay semantics after loading optimizer state.
Infra	Training–inference probability disagreement	Use inconsistent normalization constants or rotary-position layouts; break the pairing between attention outputs and their LSE merge weights.
Infra	Incorrect attention or loss support	Omit segment isolation from packed attention; exclude genuine EOS targets when padding and EOS share a token ID.
Infra	Rollouts use an unintended policy version	Skip inference-engine weight refresh; leave a captured CUDA graph bound to old parameter buffers despite a successful host-side refresh.
Infra	Precision-induced update errors	Round log-probability operands to BF16 before subtraction; accumulate gradient-norm contributions after squaring and rounding them to FP16.
Infra	History-dependent preparation overhead	Re-encode the full conversation twice at each turn to extract a short generation prefix, despite retaining the history's token IDs.
Reward	Reward gains without task gains	Introduce evaluator preferences for response length, step labels, or citation style; reward a systematically biased answer instead of the correct one.

Continued on the next page

Table 4: Representative target anomalies and candidate interventions (continued).

Domain	Target anomaly	Illustrative candidate interventions
Reward	Duplicated or misattributed feedback	Copy a terminal reward across turns; assign rewards using mutable agent slots; attach delayed feedback to its arrival row rather than its causal action.
Reward	Distillation-target misalignment	Reverse the intended KL direction; override teacher-target temperature; combine different learned EOS preferences with student-selected top-k support.
Environment	State leakage across requests or episodes	Retain a previous request's optional argument; reuse an earlier episode's KV cache or implicit tool-pagination cursor after reset.
Environment	Misbound or reordered tool results	Join concurrent results by arrival position instead of call identity; treat segment-local stream indices as a global ordering.
Environment	Premature interaction termination	Treat a retryable tool error as terminal; share one episode-level retry counter where each tool call should have its own allowance.
Environment	Inconsistent rollout-group initial conditions	Advance the environment for only one group member; clone visible state while omitting the environment's transition-randomness state.

A.4 Task packaging and expert annotation

Material selection. The task package represents the natural training scene available at its observation time, rather than a selection of files chosen for their diagnostic relevance. It preserves contemporaneous code, configurations, data-related records, metrics, rollouts, and runtime state. Observation times may fall early, midway through, or late in training. Author-side intervention records, construction notes, diagnostic analyses, reference answers, and validation controls are excluded from the agent-facing package and retained separately. A natural log entry that directly reveals abnormal behavior remains valid evidence; an author-written explanation of why that entry identifies the target cause does not. Filtering must neither reveal the intended diagnosis nor remove natural evidence that could change its interpretation.

Annotation reconciliation. Two experts independently review each package with knowledge of the target anomaly and propose Fact, Inference, and package-revision items. Fact items capture checkable observations, measurements, and implementation behavior; Inference items interpret or connect these facts into conclusions about the target anomaly. Experts assess each proposed claim against the materials in the task package, not solely against their knowledge of the construction experiment. Items agreed upon by both experts are accepted directly: Facts and Inferences enter the rubric, while revision items trigger package revision. Items are aligned by meaning rather than wording. Disagreement includes an item proposed by only one expert, conflicting formulations, or a challenge to the strength of the supporting evidence. The third expert participates in joint discussion rather than acting as a final arbiter. The three experts may refine or merge proposals before determining whether a resulting proposal has at least two supporters. Without majority support, a Fact/Inference item is omitted, but an unresolved package-revision item disqualifies the task.

Package revision. Revision items concern the adequacy of the materials and are not scored findings. They can request missing natural records or flag residual construction artifacts and answer hints for removal. Added evidence must belong to the same scene and observation boundary. Acceptance by either the original pair or the three-expert discussion triggers the same revision process. All Fact and Inference items from the current round are invalidated, including previously accepted items; the revised package receives a complete new round of independent annotation, not a selective

recheck. Finalization therefore binds the rubric to the package version actually reviewed. The task and its rubric are finalized together only when no package revisions remain pending or unresolved.

B Task Distribution by Problem Domain

Figure 8 shows the distribution of concrete scenarios within the five problem domains introduced in Section 3.1. Each task is counted once under its primary domain.

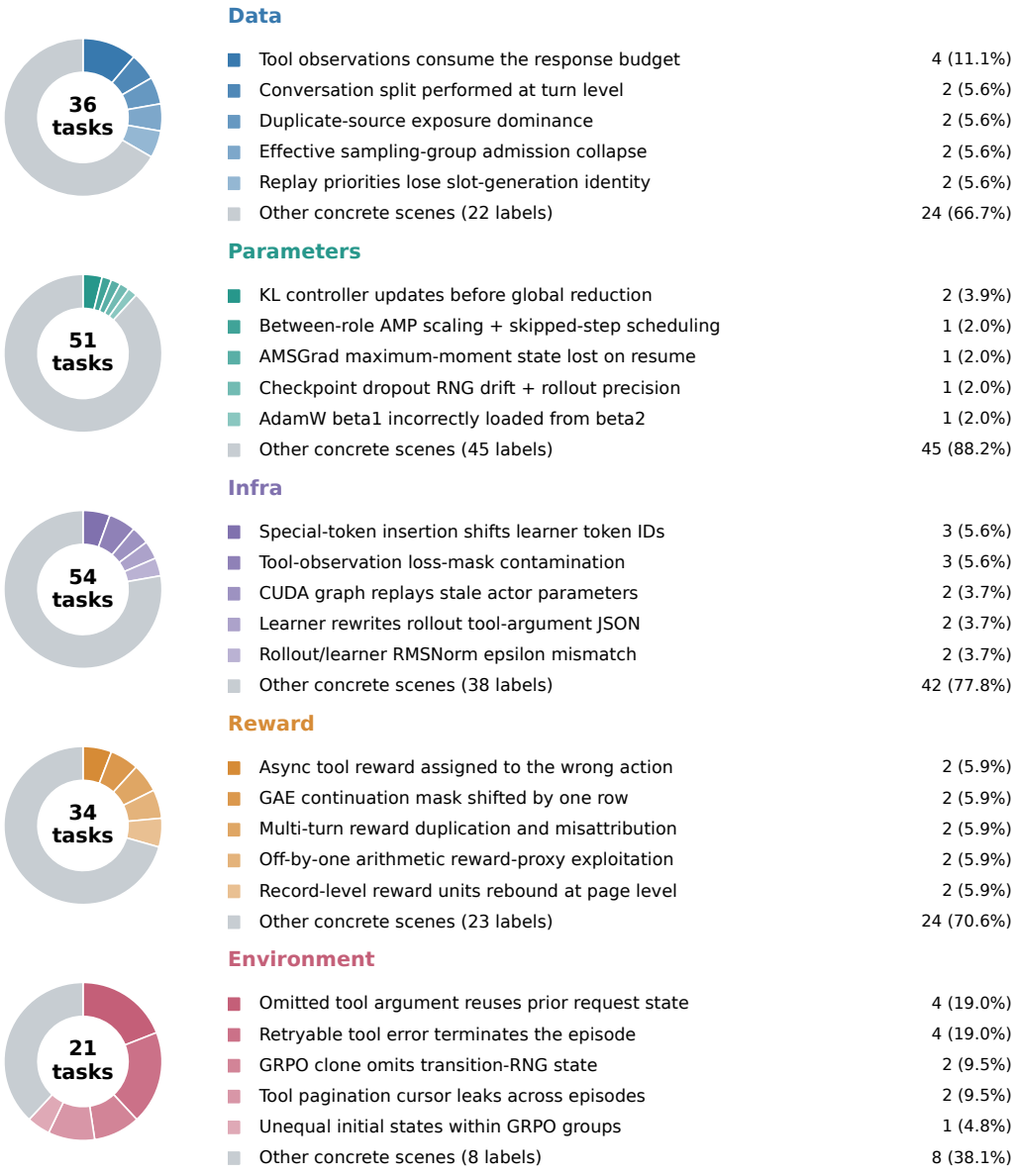


Figure 8: Composition of the 196-task benchmark. Each panel shows its five largest concrete scene groups, with the remaining groups collected as Other. Percentages use the task count of the corresponding domain as their denominator, not the full cohort. Counts refer to tasks rather than distinct scene types.

C Detailed Case Evidence

This appendix provides supplementary measurements, complete diagnostic rubrics, and repair comparisons for Section 4.2. The diagrams and derived analyses explain the cases to readers; agents receive the raw training materials instead.

C.1 Redundant Tokenization

Supplementary measurements. Figure 9 quantifies the preparation overhead described in Section 4.2.3: history length and input construction time grow across turns, while appended content stays similar in size. Panel (d) extrapolates the possible waiting cost under the stated assumptions.

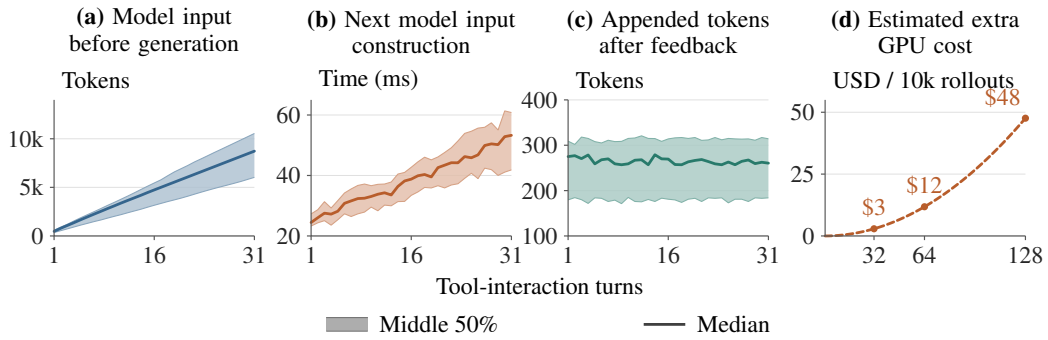


Figure 9: Measured preparation overhead (a–c) and estimated waiting cost (d). Lines/bands: medians/middle 50% across 96 trajectories reaching the 32-turn limit (31 appends), without filtering. The other 32 end earlier but also retain records. (a) Model-input tokens before action generation. (b) Next-input token construction/merge time after environment feedback, excluding environment execution and model generation. (c) Appended tokens: the observation, action list, template markers, and assistant prefix. (d) Extra H200 waiting cost over constant first-turn preparation, extrapolating (b) for 10k cumulative rollouts (64/batch, eight CPU workers, four H200 GPUs, Runpod Pods at $4.59/GPU-hour). Assumptions: balanced preparation work, all extra preparation becomes GPU waiting, and relaxed turn/context limits; not a measured bill.

Implementation detail. The generation prefix in this configuration is:

```
<|im_start|>assistant\n<think>\n\n</think>\n\n
```

The implementation obtains it by differencing two `apply_chat_template` calls on the full history. Timing records named `post_action_observation_append` cover token construction and merging after feedback arrives; they differ from `env_step_latency_seconds`, which measures environment execution. Table 5 gives the complete diagnostic rubric.

Table 5: Diagnostic criteria for redundant tokenization, paraphrased for presentation. F denotes a Fact and I an Inference.

Item	Diagnostic finding
F1	Input preparation becomes slower as interaction history grows, without a corresponding increase in the newly returned observation text.
F2	Generation-prefix IDs are obtained by encoding the complete history twice, without and with the prefix, and retaining the difference suffix.
F3	Feedback IDs are extracted by differencing encodings of short auxiliary messages that exclude the full interaction history.
F4	History token IDs are already stored in the runtime and reused directly in the next model input.
F5	The generation prefix actually appended throughout this run is the same short token sequence.
I1	Recovering the short prefix re-encodes history whose IDs are already available, then discards everything except the suffix; this redundant work grows with history length and explains the rising preparation cost.
I2	Feedback and prefix extraction use the same type of suffix difference, but only the former avoids full-history encoding through auxiliary context; the inconsistency lies in the encoded context, not the difference operation itself.

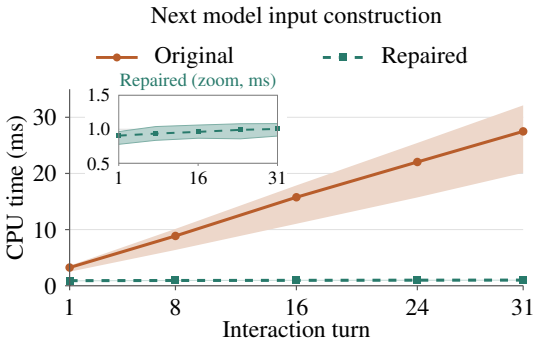


Figure 10: Tokenization repair: same-input CPU replay on 96 trajectories reaching the 32-turn limit. Points/bands show medians/middle 50% at five measured turns; each trajectory/turn uses the median of three repetitions. Timing covers next-input construction and merging, excluding environment execution and generation. The inset magnifies repaired timings. These measurements are separate from the online traces in Appendix C.1.

C.2 STOPPING DEGRADATION

Table 6 gives the complete rubric for Section 4.2.1, distinguishing observed continuation and loss support from the inferences that connect EOS preferences to the training signal.

Table 6: Diagnostic criteria for stopping degradation. F denotes a Fact and I an Inference.

Item	Diagnostic finding
F1	Some answers that stop normally before training become repetitive continuations that reach the generation limit after training.
F2	Actual generation calls accept both <|endoftext|> and <|im_end|> as stop tokens.
F3	The distillation candidate set is selected by the student's top-k; the teacher uses the same token IDs.
F4	Both distributions are separately normalized over this candidate set before reverse KL is computed.
F5	The distillation loss covers both answer content and when the answer ends, including EOS prediction.
I1	The models have different stopping preferences: the student favors <|endoftext|>, while the teacher favors <|im_end|>.
I2	The student's preferred EOS participates in the loss but receives little teacher probability because of the different stopping preference; the student-top-k reverse-KL objective may therefore suppress the student's existing stopping behavior.
I3	The teacher's preferred EOS has low student probability and is prone to exclusion from the student's top-k; when excluded, it receives no direct supervision from that position's distillation loss.
I4	Different stopping preferences, suppression of the student's existing stop, and missing direct supervision for the teacher's preferred EOS may jointly contribute to excessive continuation and failure to stop.
I5	With separate normalization only over student-top-k candidates, tokens outside the set do not enter that position's loss and receive no direct learning signal from it.

Extended repair comparison. Table 7 extends the rubric-guided repair comparison in Section 4.2.1 with two validation problems. Swapping the teacher's EOS target logits restores termination in both examples, although both mathematical answers remain incorrect.

Table 7: Actual answers before and after correcting EOS supervision. The original and repaired training runs use the same initial weights, validation questions, and sampling settings; the rows compare outputs after updates 2 and 8, respectively. Questions are summarized, and the ellipsis in the radical abbreviates denominator digits. Token counts cover the entire generated sequence, including EOS when present.

Question summary	Original training output	Corrected supervision
Complex product: $z_n = \sqrt{3/n} + i$; find $\lvert z_1 \cdots z_{47} \rvert$.	Repeats the expression $2\sqrt{\frac{240}{4031088\ldots203736}}$ until reaching the 2,048-token limit.	Gives the final answer $\frac{2}{\sqrt{168905412480}}$, then emits `<|endoftext|>` and stops. (514 tokens in total.)
Color 100 points on a circle with five colors, using each at least once. Find the minimum length guaranteeing a consecutive segment with at least three colors.	Repeats `1, 2, 3, 4, 5, 1, 2, 3, 4, 5, ...` until reaching the 2,048-token limit.	Gives the final answer $\boxed{32}$, then emits `<|endoftext|>` and stops. (489 tokens in total.)

C.3 LSE MISMATCH

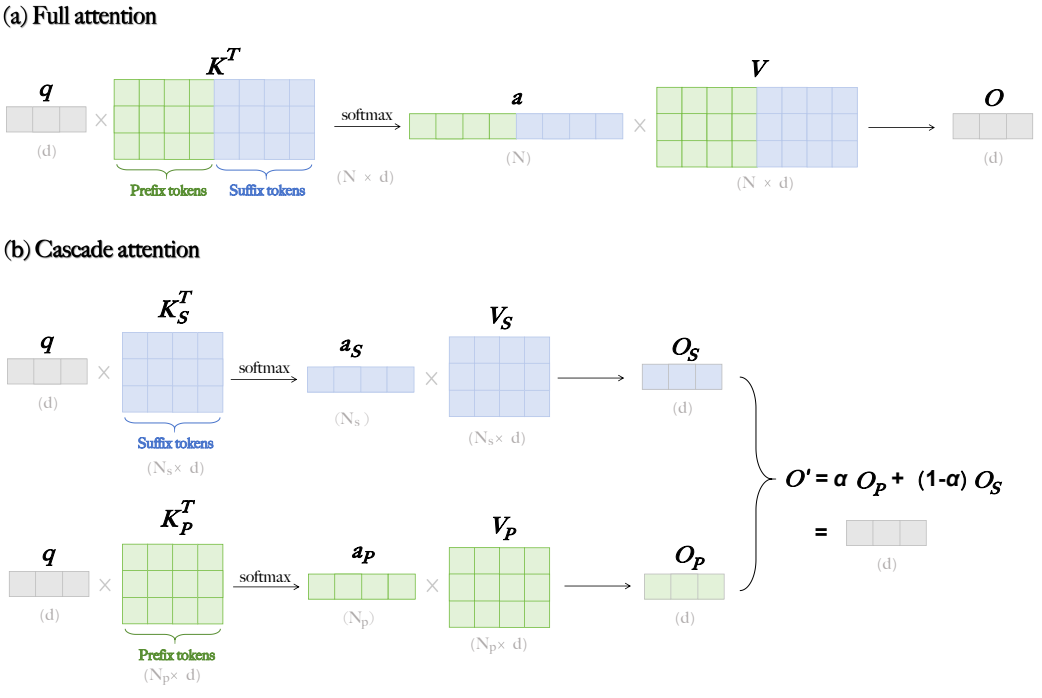


Figure 11: Full versus Cascade Attention. Separate prefix/suffix computations recover the full output when LSE values match their local outputs; incorrect pairing changes the merge weights. Blocks and weights are schematic. $N = N_P + N_S$ is the visible token count and d the head dimension; attention scaling is omitted.

Attention computation and layout. For one query head, let s_t and $\mathbf{v}_t$ denote the attention score and value vector at position t. For block $A \in \{P, S\}$, define $Z_A = \sum_{t \in A} e^{s_t}$, $L_A = \log Z_A$, and

$\mathbf{O}_A = Z_A^{-1} \sum_{t \in A} e^{s_t} \mathbf{v}_t$. Full attention over the prefix and suffix is then

$$\mathbf{O} = \frac{Z_P \mathbf{O}_P + Z_S \mathbf{O}_S}{Z_P + Z_S} = \frac{e^{L_P} \mathbf{O}_P + e^{L_S} \mathbf{O}_S}{e^{L_P} + e^{L_S}}.$$

This identity requires each LSE value to match its query head's local output. In the affected vLLM (Kwon et al., 2023) GQA path, FA2 instead interprets key/value-head, group, sample data as key/value-head, sample, group, preserving shape but misassigning the merge weights (Figure 11).

Table 8 gives the complete diagnostic rubric.

Table 8: Diagnostic criteria for the LSE mismatch, paraphrased for presentation. F denotes a Fact and I an Inference.

Item	Diagnostic finding
F1	Raw generation records contain long, repetitive digit continuations.
F2	After matching the same input and response token sequences, some tokens have inference and recomputed training log probabilities that differ by more than 1.
F3	With GQA dimension swapping enabled, the split-KV merge kernel writes LSE in key/value-head, group, sample order.
F4	FA2 rearranges the GQA-swapped LSE as though its order were key/value head, sample, group.
F5	Cascade Attention uses the prefix and suffix LSE values to determine the weights for merging their two local attention outputs.
I1	With matching weights and temperature, the two paths should assign similar probabilities under the same input prefix, within numerical error; the large observed differences therefore warrant further investigation.
I2	In the GQA-swapped split-KV merge path, misinterpreting the kernel's output layout disrupts the correspondence between LSE values and each query's attention heads.
I3	Mismatched LSE values give Cascade Attention incorrect prefix/suffix merge weights; the merged output can therefore be wrong even if the local outputs are correct.

Extended repair comparison. The maximum absolute log-probability difference falls from 12.94 to 0.723 after the rubric-guided correction, using each run's matched records rather than a fixed-sequence replay. Table 9 shows the corresponding improvements in validation outputs.

Table 9: Actual answers before and after correcting the LSE layout. The outputs are from initial validation with matching questions and decoding settings, before optimizer updates. Questions and repaired calculations are summarized; text fragments and answer lines are excerpts from the raw outputs. Ellipses elide repeated digits.

Question summary	Original output	Corrected output
Buy 16 glasses at \$5 each, with every second glass discounted to 60% of its price.	`The first 100000...` 768 tokens; reaches the limit.	$8 \times 5 + 8 \times 3 = 64$ `#### 64` 134 tokens; stops normally.
Seattle has 20 sheep; Charleston has four times as many, and Toulouse twice Charleston's number. Find the total.	`Charleston has 20 * 40 = 800000...` 768 tokens; reaches the limit.	$20 + 80 + 160 = 260$ `#### 260` 92 tokens; stops normally.