# LEARNING FROM INTERACTION EXPERIENCE VIA LOCALIZED CONTEXT DISTILLATION

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Learning from interaction experience is essential for enabling large language models to continually improve during deployment. Context distillation offers a promising path: privileged information extracted from experience is provided to a teacher, whose behavior is then distilled into a student model that has no access to it. However, both existing forms of context distillation have limitations. Off-policy context distillation trains the student on teacher-generated trajectories and thus suffers from exposure bias; whereas on-policy context distillation trains on the student's own rollouts, whose states are often uninformative and fail to convey behaviors the student cannot yet produce. To resolve this dilemma, we introduce **Localized Context Distillation (LoCD)**. LoCD first localizes the critical spans of a student trajectory that require revision. The teacher then regenerates only these spans from the student's own prefix, and the student is trained on the regenerated segments. LoCD thereby concentrates learning on the most informative states while keeping exposure bias low. Extensive experiments across mathematical reasoning, code generation, and agent tasks demonstrate that LoCD consistently outperforms strong baselines and achieves more stable training with smaller and sparser parameter updates.

## 1 INTRODUCTION

Large language models (LLMs) acquire most of their capabilities before deployment, through supervised fine-tuning on curated demonstrations and reinforcement learning with verifiable rewards (Ouyang et al., 2022; Guo et al., 2025). Once deployed, however, a model inevitably meets situations its training never covered and makes mistakes, creating a persistent need for further learning. Fortunately, each such mistake leaves behind an **interaction experience**, i.e., the model's own trajectory together with feedback from the environment or the user, which reveals where the behavior went wrong and how it can be revised (Chen et al., 2025; Wang et al., 2026b; Zhang et al., 2025). This raises a central research question: **how can LLMs learn from interaction experience so that they continually improve during deployment?**

To tackle this, a broad line of work stores experience in external memory and retrieves it as additional context when similar situations recur (Zhao et al., 2024; Chhikara et al., 2025; Hao et al., 2026). However, as interaction experience continues to accumulate, retrieving relevant experience becomes increasingly difficult, and noisy or mismatched experience may interfere with model behavior (Salama et al., 2025; Chao et al., 2026). In contrast, *Context Distillation* offers a parametric alternative that internalizes useful knowledge from experience directly into model parameters: the experience-derived context is treated as **privileged context**, available to a teacher during training but absent at inference; the same model without the context acts as the student and is trained to match the teacher's behavior (Askell et al., 2021; Snell et al., 2022; Hu et al., 2024). Existing context distillation methods mainly fall into two categories:

Off-policy context distillation strategy lets the privileged teacher generate a revised trajectory and trains the student to reproduce it (Scheurer et al., 2023; Bai et al., 2022). This provides direct supervision on the revised behavior, but it creates a mismatch between training and inference known as *exposure bias*: the student is only ever supervised on states visited by the teacher, yet at inference it must condition on states produced by its own policy. Such exposure bias can make it difficult to reproduce teacher behavior over long trajectories and may further exacerbate catastrophic forgetting (Wang et al., 2026a; Shenfeld et al., 2026).



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.

On-policy context distillation addresses this mismatch by training on the student's trajectories instead (Zhao et al., 2026a; Shenfeld et al., 2026; Ye et al., 2026b; Wang et al., 2026d): the student samples a trajectory on its own, and the privileged teacher provides token-level supervision at each student-generated prefix. Training and inference thus share the same state distribution. However, we find that restricting supervision entirely to student-generated trajectories introduces another bottleneck: on-policy context distillation readily reinforces behaviors that already appear in the student's own rollouts, but struggles to directly internalize new behaviors induced by privileged context. Empirically, as the student's initial accuracy decreases, on-policy context distillation becomes substantially less effective. On cases where the student fails across all sampled rollouts, off-policy distillation still learns substantially from the privileged context, whereas on-policy context distillation yields only marginal improvement. To probe the learning mechanism behind this limitation, we further conduct a controlled knowledge-injection experiment in which we vary the length of the answer to be learned. We observe a progressive learning pattern: longer answers require progressively more training to learn. This pattern reflects a sequential dependency: later parts become effectively learnable only after the preceding parts begin to appear in the student's own rollouts. Such sequential dependency in on-policy context distillation makes behaviors that deviate from the student's current distribution difficult to learn.

Given the weaknesses of the two paradigms, we propose a new strategy, called **Localized Context Distillation (LoCD)**. Its core idea is to introduce a short segment of teacher-generated behavior only where the student-generated trajectory requires revision. Specifically, LoCD first localizes a *critical span* of the student trajectory that requires revision and attributes the corresponding privileged context to that span. It then performs localized distillation: starting from the student's own prefix, the privileged teacher regenerates only this span, and the student is trained on the teacher-generated segment. This design places supervision on the most useful states: the span starts at the critical point where the student trajectory begins to go wrong, and the states within it are generated by the teacher, so the supervision is informative. At the same time, the teacher continues from the student's own prefix and intervenes only within this short span, so the training states stay close to the student's distribution and exposure bias remains low.

Extensive experiments across mathematical reasoning, code generation, interactive decision-making, and software-engineering tasks show that LoCD consistently outperforms strong baselines for learning from interaction experience. Compared with the strongest baselines, LoCD improves performance by 5.66% on mathematical reasoning, 2.65% on code generation, 5.52% on WebShop, and 7.68% on ALFWorld. Beyond final performance, LoCD exhibits more stable training dynamics and avoids the sustained policy-entropy collapse observed in on-policy and off-policy context distillation. Further analysis shows that these gains are achieved with substantially smaller and sparser parameter updates, supporting the effectiveness of restricting supervision to critical spans.

## 2 Preliminaries

### 2.1 Interaction Experience and Privileged Context

We consider a language model that is given a task $x$ and produces an interaction trajectory

$$\tau = (a_1, f_1, \ldots, a_K, f_K),$$

where $a_k$ denotes the model output at turn $k$, such as a reasoning step, an action, or a response, and $f_k$ denotes the feedback provided after that output. We refer to $(x, \tau)$ as an *interaction experience*. In interactive environments, the feedback $f_k$ can take the form of execution results, tool responses, or environment observations. For single-turn reasoning and coding tasks without native interaction feedback, we construct an analogous interaction experience by using a reference solution as feedback for the model output, following prior work (Kara & Ersoy, 2026; Zhao et al., 2026b; Wang et al., 2026c).

The feedback contained in an interaction experience can provide information about errors in the model's reasoning or actions and how its behavior should be improved (Scheurer et al., 2023; Hao et al., 2026; Wang et al., 2026e). Following prior work (Ye et al., 2026a; Liu et al., 2026), we use the model itself to extract a compact context from each interaction experience:

$$c \sim \mathcal{E}(\cdot \mid x, \tau),$$

where $\mathcal{E}$ denotes the experience extractor and $c$ is the resulting *privileged context*. We study how to internalize the useful information in the privileged context $c$ into model parameters, so that the

model can improve future behavior without accessing the original interaction experience at inference time.

### 2.2 CONTEXT DISTILLATION

Context distillation provides a framework for learning from interaction experience by using a teacher with access to privileged context to supervise a student without access to that context (Askell et al., 2021; Snell et al., 2022). In our setting, the experience-derived context $c$ serves as the privileged context for distillation (Hübotter et al., 2026; Liu et al., 2026; Feng et al., 2026).

Let $y = (y_1, \ldots, y_T)$ denote the trajectory used for training. We denote the student and privileged teacher by $\pi_S(\cdot \mid x, y_{<t})$ and $\pi_T(\cdot \mid x, c, y_{<t})$, respectively. In the self-distillation setting studied in this work, they share the same underlying model and differ only in whether the privileged context $c$ is provided. Formally, given a policy $\pi$, context distillation minimizes the token-level divergence between the privileged teacher and the student,

$$\mathcal{L}_{\mathrm{CD}} = \mathbb{E}_{(x,\tau)\sim\mathcal{D},\, c\sim\mathcal{E}(\cdot|x,\tau),\, y\sim\pi}\left[\frac{1}{|y|}\sum_{t=1}^{|y|} D(\mathrm{sg}[\pi_T(\cdot \mid x, c, y_{<t})]\,, \pi_S(\cdot \mid x, y_{<t}))\right].$$

Here, $\mathcal{D}$ denotes the set of interaction experiences collected online from the current student, $\mathrm{sg}[\cdot]$ denotes stop-gradient, and $D$ denotes a distributional divergence, such as forward KL, reverse KL, or Jensen–Shannon divergence (Agarwal et al., 2024; Zhao et al., 2026a). Off-policy and on-policy context distillation differ in the rollout policy $\pi$ used to generate the distillation trajectory $y$. For off-policy context distillation, $\pi = \pi_T$ and $y \sim \pi_T(\cdot \mid x, c)$, whereas for on-policy context distillation, $\pi = \pi_S$ and $y \sim \pi_S(\cdot \mid x)$.

## 3 EMPIRICAL STUDY

Recent studies find that on-policy context distillation does not consistently improve performance across a range of settings (Zhu et al., 2026; Shen et al., 2026a; Kaur et al., 2026). This leaves open whether on-policy context distillation can effectively learn behaviors induced by privileged context, especially when those behaviors are rarely or never produced by the student itself. We therefore examine how its ability to learn such behaviors varies with the student's initial accuracy, using accuracy as a measure of how frequently successful behavior appears in the student's own rollouts.

**Experimental setup.** We conduct this study on the mathematical reasoning dataset OPENTHOUGHTS (Guha et al., 2026) using three models: Qwen3-4B, Qwen3-8B, and Falcon3-10B-Instruct. We construct the interaction experiences and privileged context following Section 2, with full details provided in Appendix A.1.

For each problem, we measure the student's initial accuracy over 128 independent rollouts (avg@128) and partition problems into bins according to this accuracy. We sample 100 problems from each bin and train both methods for 5 epochs, using OPCD (Ye et al., 2026b) for on-policy context distillation and TRD (Jiang et al., 2026) for off-policy context distillation. After training, we evaluate the distilled student without privileged context on the same problems used for training.

Let $P_{\mathrm{base}}$ and $P_T$ denote the performance of the initial student and the privileged teacher, respectively. We define the additional performance enabled by privileged context as the *privileged improvement*, $\Delta_{\mathrm{priv}} = P_T - P_{\mathrm{base}}$. We compare the distilled student's improvement over $P_{\mathrm{base}}$ with $\Delta_{\mathrm{priv}}$ to measure how effectively it learns from the privileged context.

**On-policy context distillation becomes less effective at low initial accuracy.** As shown in Figure 1, all three models exhibit a consistent pattern: the effectiveness of on-policy context distillation decreases as the student's initial accuracy becomes lower. When the student's initial accuracy is above $60\%$, on-policy context distillation retains a substantial portion of the privileged improvement and outperforms off-policy context distillation. At lower initial accuracy levels, however, this advantage disappears, and on-policy context distillation becomes substantially less effective than off-policy context distillation.

The gap between the two methods is most pronounced for problems with zero initial accuracy (avg@128 $= 0$). Although the initial student produces no correct output among 128 independent

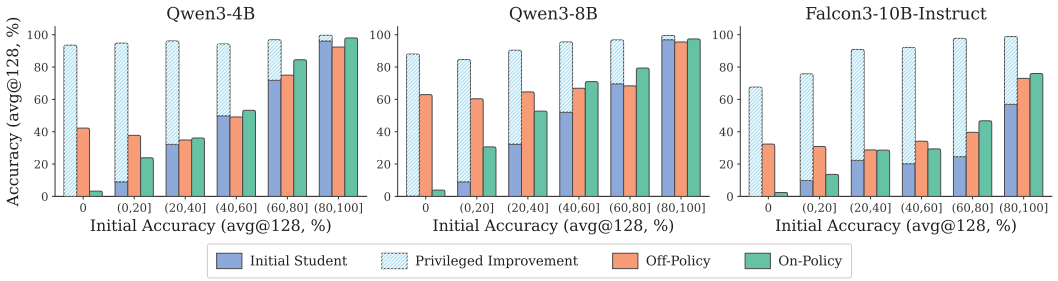



Figure 1: How initial student accuracy affects learning from privileged context.

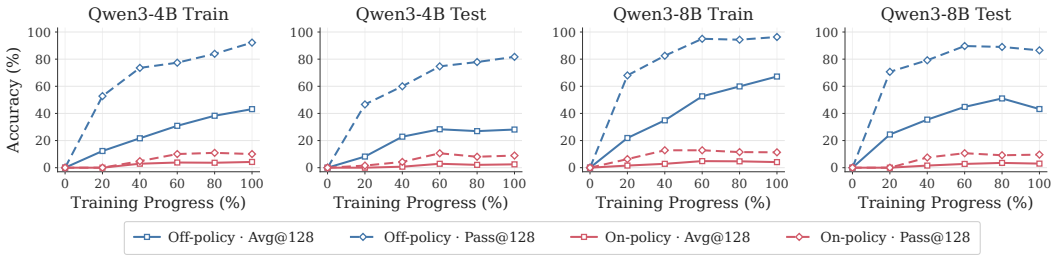



Figure 2: Training dynamics on problems with zero initial accuracy (avg@128 = 0).

rollouts, the privileged teacher still substantially outperforms the initial student, showing that the privileged context provides useful information for these problems. This provides a clean setting for testing whether the two distillation methods can learn from useful privileged context when the desired behavior is absent from the student's own rollouts. As shown, off-policy context distillation substantially improves the student, whereas on-policy context distillation yields only marginal improvement.

We further examine whether this gap persists with continued training. Figure 1 tracks avg@128 and pass@128 throughout training on problems with zero initial accuracy. For both models, off-policy context distillation improves rapidly early in training and continues to achieve substantially higher avg@128 and pass@128. In contrast, on-policy context distillation shows only limited improvement, and the gap between the two methods persists throughout training. To examine whether this pattern extends beyond the training problems, we construct a held-out set of 100 problems with zero initial accuracy. The same dynamics emerge on the held-out set: off-policy context distillation improves substantially, whereas on-policy context distillation remains far below it throughout training.

Across these experiments, on-policy context distillation becomes less effective when behaviors induced by privileged context rarely appear in the student's own rollouts. This phenomenon points to a simple and important mechanism: on-policy context distillation applies teacher supervision only on student-generated prefixes, which creates a sequential dependency: later parts of a new behavior become effectively learnable only after the preceding parts begin to appear in the student's own rollouts. Consequently, longer target behaviors should require progressively more training to learn.

**On-policy context distillation exhibits length-dependent learning.** To further examine this explanation, we construct a controlled knowledge-injection task following prior work (Chang et al., 2024; Zhang et al., 2026b). Each example consists of a synthetic fact and a corresponding question, such as "*Sne's coach is Cour*" and "*What is Sne's coach?*" The synthetic fact is directly used as privileged context during training.

We vary the length of the answer to be learned, with $L \in \{4, 8, 16, 32\}$ tokens, and mix all length groups within the same training run. We use the same on-policy and off-policy context distillation methods as above, training for 200 epochs and evaluating every 5 epochs. Full experimental details are provided in Appendix A.2.

As shown in Figure 3, off-policy context distillation rapidly learns answers of all lengths, with little sensitivity to answer length. In contrast, on-policy context distillation exhibits a clear length-dependent delay: performance begins to improve earliest for $L = 4$, followed by $L = 8$, $L = 16$,

and $L = 32$. Moreover, the $L = 32$ group remains below the shorter-length groups under the same training budget.

This length-dependent delay supports the sequential dependency described above. Although the complete answer is available to the privileged teacher from the beginning, later parts become effectively learnable only after the preceding parts begin to appear in the student's own rollouts.

These findings are particularly relevant to learning from failed interaction trajectories. Such trajectories arise precisely when the current policy does not yet produce the desired behavior, making the required correction less likely to appear in the student's own rollouts. This places learning from failed interactions in the regime where on-policy context distillation is less effective.

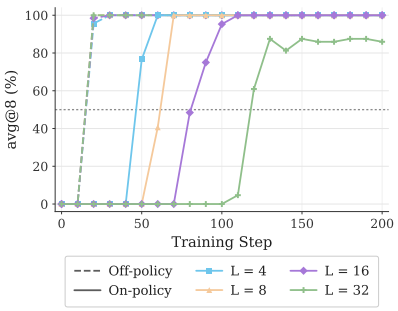



Figure 3: Knowledge-injection dynamics across answer lengths.

## 4 Localized Context Distillation

Our empirical study shows that on-policy context distillation becomes much less effective when the behaviors induced by privileged context rarely appear in the student's own rollouts. The key limitation is that privileged supervision depends on student-generated prefixes, making behaviors that the current policy cannot reliably produce difficult to incorporate into training.

A natural alternative is to let the privileged teacher directly generate a revised trajectory, as in off-policy context distillation. While this can explicitly introduce behaviors absent from the student's own rollouts, training on teacher-generated states introduces exposure bias, which may further contribute to catastrophic forgetting or reasoning collapse (Shenfeld et al., 2026; Kim et al., 2026).

To address the limitations of both approaches, we introduce *Localized Context Distillation* (LoCD). As illustrated in Figure 4, LoCD preserves the student-generated prefix and introduces privileged teacher generation only within an attributed *critical span*. Specifically, LoCD consists of two steps: *localization and attribution* and *localized distillation*. After each update, the resulting student policy is used to collect the next round of interaction experience.

**Localization and attribution.** Given an interaction experience $(x, \tau)$, LoCD localizes the spans that require revision and extracts their corresponding privileged contexts:

$$([\alpha, \beta), c) \sim \mathcal{A}(\cdot \mid x, \tau).$$

Here, $\mathcal{A}$ is a prompt-based localization and attribution procedure implemented by prompting the same model as the student, leveraging its own reasoning capabilities. The indices $\alpha$ and $\beta$ denote the start and end indices of an attributed *critical span* along the student trajectory, and $c$ denotes its corresponding privileged context. For example, a critical span can correspond to a contiguous reasoning segment or a specific interaction step in an agent trajectory. When multiple critical spans are identified from the same interaction, we treat each attributed span as a separate training instance.

**Localized distillation.** Given an attributed critical span $([\alpha, \beta), c)$, LoCD samples a local sequence $y^T$ from the privileged teacher starting from the student-generated prefix $y^S_{<\alpha}$. We optimize

$$\mathcal{L} = \mathbb{E}_{\substack{(x,\tau)\sim\mathcal{D},\ ([\alpha,\beta),c)\sim\mathcal{A}(\cdot|x,\tau) \\ y^T\sim\pi_T(\cdot|x,c,y^S_{<\alpha})}} \left[ \frac{1}{\beta-\alpha} \sum_{t=1}^{\beta-\alpha} D\big(\mathrm{sg}\big[\pi_T(\cdot \mid x, c, y^S_{<\alpha}, y^T_{<t})\big]\,, \pi_S(\cdot \mid x, y^S_{<\alpha}, y^T_{<t})\big) \right].$$

Here, teacher sampling is truncated after position $\beta$, restricting privileged generation to the attributed critical span. The divergence $D$ can be forward KL, reverse KL, or Jensen–Shannon divergence (Agarwal et al., 2024; Zhao et al., 2026a). Following prior works (Zhao et al., 2026a; Yang et al., 2026), we clip excessively large per-token divergences for training stability.

Overall, LoCD restricts the effect of privileged context to the attributed *critical span*. Unlike off-policy context distillation, where privileged context can reshape the training trajectory, and on-policy context distillation, where privileged supervision is applied token by token along fixed student-generated prefixes, LoCD instead adopts a *span-level* formulation of context distillation, where teacher generation and supervision are both restricted to the attributed critical span. This design provides several advantages as follows.

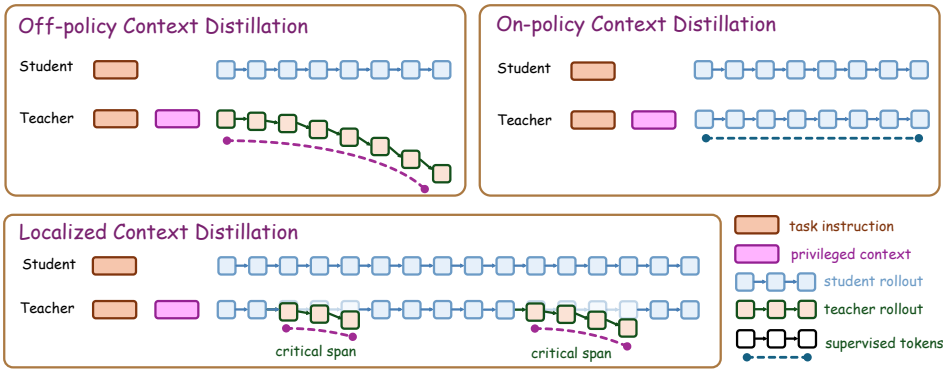



Figure 4: Illustrative comparison of off-policy, on-policy, and localized context distillation.

**Advantages of LoCD.** First, LoCD anchors each correction at a state actually visited by the current student, while allowing the privileged teacher to directly supervise behaviors that the student cannot yet produce. This enables corrective supervision directly where the student requires revision. Second, by restricting teacher-generated prefixes to the critical span, LoCD confines teacher-induced deviation to the region that requires correction, thereby limiting exposure bias. Third, through localization and attribution, LoCD matches each critical span with its corresponding privileged context. This allows the supervision induced by each privileged context to be applied to the student states where that context is semantically relevant.

## 5 Experiments

We evaluate LoCD from three main perspectives. First, we compare it with strong baselines across reasoning, coding, and agent tasks. Second, we examine its training dynamics, including downstream performance, policy entropy, and parameter updates throughout training. Third, we study key design choices and robustness to different distillation objectives.

### 5.1 Experimental Setup

**Tasks and models.** We evaluate LoCD across four settings: mathematical reasoning, code generation, interactive decision-making, and software-engineering agents. For mathematical reasoning, we train on a 28.8K subset of OpenThoughts (Guha et al., 2026); for code generation, we use Codeforces problems following the competitive-programming setting of CodeContests (Li et al., 2022); for interactive decision-making, we consider WebShop (Yao et al., 2022) and ALFWorld (Shridhar et al., 2020); and for software engineering, we use SWE-Gym (Pan et al., 2024), with additional evaluation on SWE-bench Verified. Our main experiments use Qwen3-8B (Yang et al., 2025a) in non-thinking mode. To examine whether the gains generalize across model families, we additionally evaluate Falcon3-10B-Instruct (Team, 2024) on mathematical reasoning. For repository-level software engineering, we use Qwen3-Coder-30B-A3B-Instruct (Yang et al., 2025a).

**Baselines.** We compare LoCD with representative approaches for learning from interaction experience. Text2Grad (Wang et al., 2026b) converts free-form language feedback into span-level learning signals for policy optimization. TRD (Jiang et al., 2026) uses teacher guidance to revise student-generated trajectories before distillation, providing a strong off-policy alternative. For on-policy context distillation, OPCD (Ye et al., 2026b) distills a context-conditioned teacher on student-generated trajectories; ROSD (Zhao et al., 2026b) identifies the first erroneous region and restricts distillation to the affected suffix; and OP$^2$SD (Ichihara et al., 2026) uses unrelated privileged solutions to induce alternative teacher behavior while retaining student-generated rollouts. For long-horizon agent tasks, we additionally compare with Hint-SD (Yeo et al., 2026), which applies targeted hindsight supervision to failure-relevant parts of an agent trajectory.

### 5.2 Main Results

Tables 1 and 2 show that LoCD consistently outperforms the baselines on both math reasoning and coding tasks. On Qwen3-8B, LoCD achieves the best result across all eight mathematics benchmarks, improving the average score from 42.39 to 48.05; the same advantage holds on Falcon3-10B-Instruct. For code generation, LoCD leads across all Codeforces difficulty ranges and LiveCodeBench, reaching an average score of 17.20 versus 14.55 for the strongest baseline.

| Method | AIME24 | AIME25 | HMMT25-Feb | HMMT25-Nov | Olympiad | MATH-500 | Minerva | AMO | Avg. |
|---|---|---|---|---|---|---|---|---|---|
| *Qwen3-8B-Base* | | | | | | | | | |
| Base | 28.06 | 20.83 | 11.11 | 11.67 | 55.34 | 85.80 | 39.34 | 2.33 | 31.81 |
| Text2Grad | 46.32 $\pm$ 3.7 | 33.51 $\pm$ 2.4 | 21.08 $\pm$ 1.2 | 28.56 $\pm$ 1.9 | 60.24 $\pm$ 2.6 | 91.09 $\pm$ 1.1 | 38.70 $\pm$ 0.9 | 4.27 $\pm$ 0.6 | 40.47 |
| TRD | 48.00 $\pm$ 4.1 | 37.50 $\pm$ 1.2 | 21.39 $\pm$ 1.3 | 26.80 $\pm$ 1.0 | 60.69 $\pm$ 3.1 | 89.31 $\pm$ 0.7 | 40.26 $\pm$ 0.8 | 4.75 $\pm$ 0.4 | 41.09 |
| OPCD | 54.23 $\pm$ 2.8 | 38.19 $\pm$ 1.8 | 21.37 $\pm$ 1.6 | 29.16 $\pm$ 4.3 | 60.91 $\pm$ 1.7 | 89.35 $\pm$ 1.8 | 40.81 $\pm$ 1.6 | 5.08 $\pm$ 0.6 | 42.39 |
| OP$^2$SD | 56.87 $\pm$ 2.7 | 39.02 $\pm$ 2.4 | 20.29 $\pm$ 1.4 | 26.90 $\pm$ 2.0 | 60.42 $\pm$ 2.6 | 90.04 $\pm$ 1.1 | 40.06 $\pm$ 2.2 | 4.72 $\pm$ 0.7 | 42.29 |
| ROSD | 48.69 $\pm$ 4.4 | 36.57 $\pm$ 2.0 | 23.25 $\pm$ 1.6 | 24.63 $\pm$ 1.4 | 61.95 $\pm$ 2.2 | 87.87 $\pm$ 0.8 | 39.15 $\pm$ 1.2 | 4.56 $\pm$ 0.4 | 40.83 |
| **LoCD** | 61.34 $\pm$ 3.6 | 47.91 $\pm$ 4.6 | 25.28 $\pm$ 2.5 | 38.47 $\pm$ 3.0 | 66.32 $\pm$ 3.0 | 93.78 $\pm$ 1.5 | 44.65 $\pm$ 2.6 | 6.67 $\pm$ 1.0 | 48.05 |
| *Falcon3-10B-Instruct* | | | | | | | | | |
| Base | 4.44 | 3.06 | 0.56 | 4.44 | 22.55 | 51.40 | 32.35 | 2.33 | 15.16 |
| Text2Grad | 9.12 $\pm$ 1.4 | 3.30 $\pm$ 0.7 | 1.39 $\pm$ 0.6 | 5.00 $\pm$ 0.7 | 35.29 $\pm$ 1.7 | 64.83 $\pm$ 0.8 | 34.57 $\pm$ 2.4 | 3.02 $\pm$ 0.5 | 19.57 |
| TRD | 11.11 $\pm$ 1.6 | 5.56 $\pm$ 0.8 | 2.50 $\pm$ 0.7 | 4.44 $\pm$ 0.6 | 37.69 $\pm$ 1.8 | 72.13 $\pm$ 0.9 | 38.60 $\pm$ 2.5 | 3.33 $\pm$ 0.6 | 21.92 |
| OPCD | 10.83 $\pm$ 1.4 | 6.39 $\pm$ 0.7 | 1.67 $\pm$ 0.6 | 5.28 $\pm$ 0.7 | 35.31 $\pm$ 1.7 | 69.67 $\pm$ 0.9 | 38.60 $\pm$ 2.3 | 2.83 $\pm$ 0.6 | 21.32 |
| OP$^2$SD | 10.52 $\pm$ 1.5 | 5.28 $\pm$ 0.7 | 2.22 $\pm$ 0.6 | 5.00 $\pm$ 0.7 | 36.35 $\pm$ 1.7 | 70.53 $\pm$ 0.9 | 37.74 $\pm$ 2.3 | 3.17 $\pm$ 0.5 | 21.35 |
| ROSD | 10.72 $\pm$ 1.3 | 6.33 $\pm$ 0.6 | 1.39 $\pm$ 0.5 | 4.06 $\pm$ 0.6 | 33.23 $\pm$ 1.7 | 68.73 $\pm$ 0.8 | 39.34 $\pm$ 2.1 | 3.17 $\pm$ 0.5 | 20.87 |
| **LoCD** | 12.39 $\pm$ 1.7 | 7.22 $\pm$ 0.9 | 3.13 $\pm$ 0.7 | 5.56 $\pm$ 0.8 | 39.91 $\pm$ 1.8 | 73.47 $\pm$ 1.0 | 40.44 $\pm$ 2.6 | 3.84 $\pm$ 0.6 | 23.25 |

Table 1: Performance on mathematical reasoning benchmarks.

| Method | CF$_{800\text{-}1200}$ | CF$_{1200\text{-}1600}$ | CF$_{1600+}$ | LCB | Avg. |
|---|---|---|---|---|---|
| *Qwen3-8B* | | | | | |
| Base | 26.47 | 8.20 | 0.55 | 16.71 | 12.98 |
| Text2Grad | 24.02 $\pm$ 1.2 | 8.03 $\pm$ 1.0 | 0.37 $\pm$ 0.1 | 16.71 $\pm$ 0.8 | 12.28 |
| TRD | 28.66 $\pm$ 1.6 | 10.69 $\pm$ 1.3 | 0.65 $\pm$ 0.1 | 18.19 $\pm$ 0.8 | 14.55 |
| ROSD | 28.09 $\pm$ 1.5 | 9.68 $\pm$ 2.1 | 0.62 $\pm$ 0.2 | 17.10 $\pm$ 1.1 | 13.87 |
| OPCD | 27.20 $\pm$ 1.8 | 9.88 $\pm$ 0.5 | 0.70 $\pm$ 0.2 | 17.07 $\pm$ 0.2 | 13.71 |
| OP$^2$CD | 25.36 $\pm$ 1.4 | 8.92 $\pm$ 0.6 | 0.70 $\pm$ 0.2 | 17.07 $\pm$ 0.2 | 13.01 |
| **LoCD** | 30.94 $\pm$ 1.1 | 12.91 $\pm$ 1.5 | 0.79 $\pm$ 0.2 | 24.14 $\pm$ 2.1 | 17.20 |

Table 2: Performance on code-generation benchmarks.

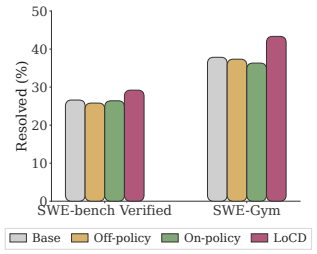



Figure 5: Performance on SWE.

Beyond single-turn reasoning and coding tasks, LoCD remains superior in multi-turn agent tasks. LoCD consistently improves long-horizon agent performance. As shown in Table 3, LoCD outperforms the strongest baselines by 5.52 points on WebShop and 7.68 points on ALFWorld. Figure 5 shows the same trend on SWE-bench Verified and SWE-Gym, where LoCD achieves the highest resolved rate among the compared methods.

### 5.3 Training Dynamics

**Performance over training.** Figure 6 compares LoCD with on-policy and off-policy context distillation throughout training. Across all four representative tasks, LoCD continues to improve as training progresses, whereas the two baselines tend to plateau or fluctuate. The separation is particularly pronounced on WebShop and ALFWorld, where LoCD continues to gain performance after the baselines have largely saturated.

**Policy entropy.** Figure 7 records how the output distribution evolves during training. Across OpenThoughts, Codeforces, and SWE-Gym, both on-policy and off-policy context distillation progressively reduce policy entropy. In contrast, LoCD avoids this sustained entropy collapse and maintains a more stable entropy level throughout training. Together with its stronger downstream performance, this indicates that LoCD can improve task behavior without excessively concentrating the model's output distribution.

### 5.4 Ablation Study

**Robustness to divergence choice.** We first examine whether the effectiveness of LoCD depends on the choice of distillation divergence. Figure 8 compares forward and reverse KL on AIME24, LiveCodeBench v6, and WebShop. Although the relative performance of the on-policy and off-policy baselines varies with the divergence direction, LoCD consistently achieves the strongest performance under both choices. This indicates that the effectiveness of LoCD is not specific to a particular KL direction.

| Method | WebShop (official test) | | | | ALFWorld (valid_unseen) | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | $C_1$ | $C_2$ | $C_{3+}$ | All | Pick | Look | Clean | Heat | Cool | Pick2 | All |
| Base | 58.27 | 9.04 | 9.13 | 18.03 | 27.81 | 27.83 | 18.35 | 31.92 | 27.06 | 2.03 | 22.94 |
| TRD | 63.42 | 43.31 | 30.64 | 43.40 | 15.34 | 20.42 | 19.71 | 21.43 | 21.62 | 9.83 | 18.36 |
| OPCD | 62.13 | 15.72 | 12.43 | 23.24 | 29.21 | 24.16 | 20.43 | 29.04 | 28.62 | 14.63 | 24.53 |
| ROSD | 62.31 | 47.32 | 27.14 | 44.36 | 27.05 | 25.41 | 23.92 | 31.73 | 29.64 | 13.02 | 25.53 |
| Hint-SD | 65.91 | 48.93 | 30.52 | 46.83 | 25.07 | 24.12 | 24.06 | 18.85 | 27.91 | 15.73 | 22.90 |
| **LoCD** | 69.21 | 56.73 | 33.14 | 52.35 | 33.31 | 35.24 | 26.93 | 49.31 | 36.52 | 16.51 | 33.21 |

Table 3: Performance on interactive decision-making benchmarks.

**Effect of correction-point selection.** We next examine whether feedback-guided localization is necessary. Figure 9 compares LoCD with variants that replace its feedback-selected correction points with alternative selection or training strategies. LoCD consistently performs best across all three benchmarks, while random-point, raw-trajectory, and full-train variants all lose part of its advantage. These results support the importance of localizing where revision is needed before applying teacher supervision.

## 6 Related Works

**Non-Parametric Learning from Experience.** A broad line of work reuses past interaction experience without updating model parameters. Some methods directly store and retrieve past trajectories, feedback, or reflections as in-context examples (Shinn et al., 2023; Zhao et al., 2024; Liu et al., 2025). More recent methods abstract accumulated interactions into reusable insights, guidelines, skills, or workflows that generalize across related tasks (Wang et al., 2023; Fu et al., 2024; Wang et al., 2024; Yang et al., 2025b). These methods rely on retrieving and reintroducing relevant experience at inference time, whereas we study how to internalize such experience directly into model parameters (Liu et al., 2025; Wang et al., 2024; Cao et al., 2026).

**Reinforcement Learning from Interaction Experience.** Reinforcement learning has become a central paradigm for improving large language models, typically optimizing scalar rewards derived from preference models or outcome verifiers (Ouyang et al., 2022; Shao et al., 2024). However, interaction feedback often contains richer diagnostic and corrective information expressed in natural language than scalar rewards can capture (Wang et al., 2026b; Zhou et al., 2025; Wang et al., 2026f). Recent work therefore incorporates language feedback into reinforcement learning by converting critiques into token-, span-, or process-level rewards, or by using feedback to guide refinement and exploration (Cao et al., 2024; Wang et al., 2026b; Zhang et al., 2025; Li et al., 2025). Related approaches improve credit assignment by localizing optimization to critical steps or error regions (Li et al., 2026a; Liang et al., 2026; Li et al., 2026b). Despite these advances, natural-language feedback is still largely mediated through reward or credit-assignment signals, requiring the model to discover the corresponding corrective behavior through policy optimization.

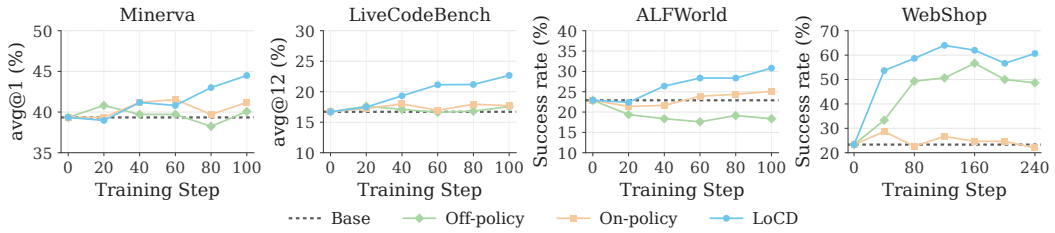



Figure 6: Training dynamics across four representative tasks.

**Context Distillation.** Context distillation internalizes behaviors induced by additional context into model parameters by training a student without that context to match a context-conditioned teacher (Askell et al., 2021; Snell et al., 2022). In learning from interaction feedback, off-policy approaches typically use privileged information to construct improved responses and then train the student on these corrections (Scheurer et al., 2023). Recent on-policy methods instead sample trajectories from

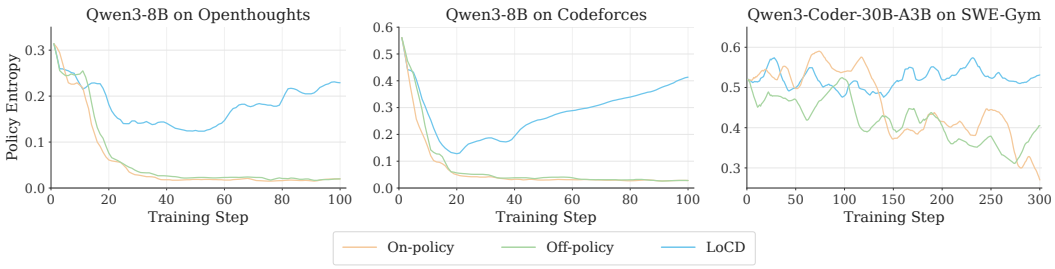



Figure 7: Policy entropy during training on OpenThoughts, Codeforces, and SWE-Gym.

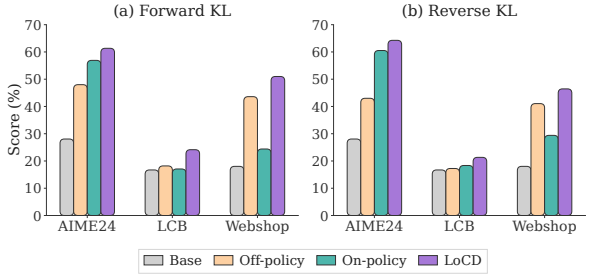



Figure 8: Robustness to divergence choice.

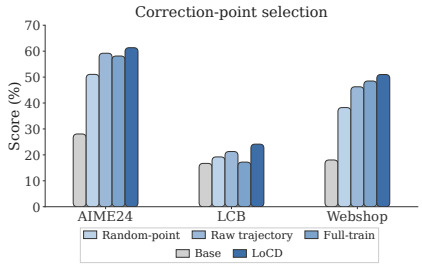



Figure 9: Effect of correction-point selection.

the current student and provide privileged teacher supervision on student-generated prefixes, allowing reference solutions, demonstrations, or textual feedback to be internalized while remaining aligned with the student's state distribution (Zhao et al., 2026a; Shenfeld et al., 2026; Hübotter et al., 2026; Ye et al., 2026b). More recent work extends this paradigm toward self-evolving models by deriving the privileged context itself from interaction experience: transferable experience can be progressively extracted from past trajectories, completed agent trajectories can be summarized into hindsight skills, and relevant experience can even be encoded into learnable latent context (Ren et al., 2026; Wu et al., 2026; Zhang et al., 2026a). Other work further improves the alignment between privileged feedback and student states by constructing locally relevant hindsight feedback or targeting failure-relevant actions for distillation (Liu et al., 2026; Yeo et al., 2026; Feng et al., 2026). Despite these advances, such methods largely retain student-generated states when applying privileged supervision. LoCD instead uses interaction feedback to identify where the student trajectory should be corrected and introduces explicit off-policy supervision only around these localized states.

## 7 Conclusion

We study how language models can internalize knowledge and corrections from interaction experience through context distillation. Our empirical study shows that on-policy context distillation becomes inefficient when the behaviors induced by privileged context are difficult for the current student to produce, with new information being acquired only progressively along student-generated trajectories. Motivated by this limitation, we introduce LoCD, which localizes where corrections are needed and introduces short teacher-generated corrective segments while preserving the surrounding student trajectory. More broadly, LoCD offers a promising path toward enabling language models to continually self-evolve from their own interaction experience in real-world environments.

## AI Use Statement

We used generative AI tools only for language polishing. All such use took place after the research had been completed and the manuscript had been written by the authors, and was limited to improving the grammar, wording and readability of text that the authors had already drafted. No part of this paper was generated from scratch by an AI tool.

We did not use generative AI tools to formulate the research idea, to design LoCD or any of its components, to write or debug the code, or to run, analyse and interpret the experiments. Every number reported in this paper comes from our own implementation and our own evaluation pipeline.

The authors reviewed all AI-assisted edits. Suggested revisions were accepted or rejected sentence by sentence, and every accepted revision was checked to confirm that it left the technical content, the claims and the scope of those claims unchanged; edits that altered, weakened or overstated a claim were discarded. The final text was then re-read by the authors for factual accuracy and internal consistency. We take full responsibility for the entire content of this paper, including every sentence that was edited with the aid of generative AI.

## REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *International Conference on Learning Representations*, volume 2024, pp. 21246–21263, 2024.

Amanda Askell, Yuntao Bai, Anna Chen, Dawn Drain, Deep Ganguli, Tom Henighan, Andy Jones, Nicholas Joseph, Ben Mann, Nova DasSarma, et al. A general language assistant as a laboratory for alignment. *arXiv preprint arXiv:2112.00861*, 2021.

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

Meng Cao, Lei Shu, Lei Yu, Yun Zhu, Nevan Wichers, Yinxiao Liu, and Lei Meng. Enhancing reinforcement learning with dense rewards from language model critic. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 9119–9138, 2024.

Zouying Cao, Jiaji Deng, Li Yu, Weikang Zhou, Zhaoyang Liu, Bolin Ding, and Hai Zhao. Remember me, refine me: A dynamic procedural memory framework for experience-driven agent evolution. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 16803–16822, 2026.

Hoyeon Chang, Jinho Park, Seonghyeon Ye, Sohee Yang, Youngkyung Seo, Du-Seong Chang, and Minjoon Seo. How do large language models acquire factual knowledge during pretraining? *Advances in neural information processing systems*, 37:60626–60668, 2024.

Hanxiang Chao, Yihan Bai, Rui Sheng, Tianle Li, and Yushi Sun. Stale: Can llm agents know when their memories are no longer valid? *arXiv preprint arXiv:2605.06527*, 2026.

Zizhao Chen, Mustafa Omer Gul, Yiwei Chen, Gloria Geng, Anne Wu, and Yoav Artzi. Retrospective learning from interactions. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 24580–24606, 2025.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready ai agents with scalable long-term memory. *arXiv preprint arXiv:2504.19413*, 2025.

Yangyang Feng, Zhuoyan Feng, and Junlan Chen. Past: Privileged adaptation from complete student trajectories for on-policy self-distillation. *arXiv preprint arXiv:2608.08726*, 2026.

Yao Fu, Dong-Ki Kim, Jaekyeom Kim, Sungryull Sohn, Lajanugen Logeswaran, Kyunghoon Bae, and Honglak Lee. Autoguide: Automated generation and selection of context-aware guidelines for large language model agents. *Advances in Neural Information Processing Systems*, 37: 119919–119948, 2024.

Etash Guha, Ryan Marten, Sedrick Keh, Negin Raoof, Georgios Smyrnis, Hritik Bansal, Marianna Nezhurina, Jean Mercat, Trung Vu, Zayne Sprague, et al. Openthoughts: Data recipes for reasoning models. In *International Conference on Learning Representations*, volume 2026, pp. 108059–108130, 2026.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, 2025.

Zhezheng Hao, Hong Wang, Jian Luo, Jianqing Zhang, Yuyan Zhou, Qiang Lin, Can Wang, Hande Dong, and Jiawei Chen. Recreate: Reasoning and creating domain agents driven by experience. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 31018–31046, 2026.

Chi Hu, Yimin Hu, Hang Cao, Tong Xiao, and Jingbo Zhu. Teaching language models to self-improve by learning from language feedback. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 6090–6101, 2024.

Jonas Hübotter, Frederike Lübeck, Lejs Behric, Anton Baumann, Marco Bagatella, Daniel Marta, Ido Hakimi, Idan Shenfeld, Thomas Kleine Buening, Carlos Guestrin, et al. Reinforcement learning via self-distillation. *arXiv preprint arXiv:2601.20802*, 2026.

Yuki Ichihara, Naoto Iwase, Mohammad Atif Quamar, and Junpei Komiyama. Privileged solutions or context-induced teacher behavior? dissecting on-policy self-distillation. *arXiv preprint arXiv:2608.09228*, 2026.

Li Jiang, Haoran Xu, Yichuan Ding, and Amy Zhang. Trajectory-refined distillation. *arXiv preprint arXiv:2606.08432*, 2026.

Semih Kara and Oğuzhan Ersoy. The role of feedback alignment in self-distillation. *arXiv preprint arXiv:2606.11173*, 2026.

Simran Kaur, Narutatsu Ri, Yinghui He, Liam Fowl, and Sanjeev Arora. Rethinking on-policy self-distillation for thinking models. *arXiv preprint arXiv:2607.05184*, 2026.

Jeonghye Kim, Xufang Luo, Minbeom Kim, Sangmook Lee, Dohyung Kim, Jiwon Jeon, Dongsheng Li, and Yuqing Yang. Why does self-distillation (sometimes) degrade the reasoning capability of llms? *arXiv preprint arXiv:2603.24472*, 2026.

Ang Li, Yifei Wang, Zhihang Yuan, Stefanie Jegelka, and Yisen Wang. Lanpo: Bootstrapping language and numerical feedback for reinforcement learning in llms. *arXiv preprint arXiv:2510.16552*, 2025.

Mukai Li, Qingcheng Zeng, Tianqing Fang, Zhenwen Liang, Linfeng Song, Qi Liu, Haitao Mi, and Dong Yu. Verified critical step optimization for llm agents. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 39627–39639, 2026a.

Yugu Li, Zehong Cao, Jianglin Qiao, and Siyi Hu. Ssvpo: Effective step-level credit assignment for rl training of language models. In *The Fourteenth International Conference on Learning Representations*, 2026b.

Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, et al. Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097, 2022.

Qiao Liang, Yuke Zhu, Chao Ge, Lei Yang, Ying Shen, Bo Zheng, and Sheng Guo. Learning from the irrecoverable: Error-localized policy optimization for tool-integrated llm reasoning. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11008–11028, 2026.

Haoran Liu, Yuwei Zhang, Xiyao Li, Bohan Lyu, and Jingbo Shang. Hero: Hindsight-enhanced reflection from environment observations for agentic self-distillation. *arXiv preprint arXiv:2606.11559*, 2026.

Yitao Liu, Chenglei Si, Karthik R Narasimhan, and Shunyu Yao. Contextual experience replay for self-improvement of language agents. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 14179–14198, 2025.

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

Jiayi Pan, Xingyao Wang, Graham Neubig, Navdeep Jaitly, Heng Ji, Alane Suhr, and Yizhe Zhang. Training software engineering agents and verifiers with swe-gym. *arXiv preprint arXiv:2412.21139*, 2024.

Shijie Ren, Xiting Wang, Meng Li, Yujie Guo, Yunhang Yao, Ziheng Peng, Xunlong Wang, Yuetan Chen, Haoyang Zhou, Yunlong Liang, et al. Self-improving large language models via progressive experience evolution. *arXiv preprint arXiv:2608.02139*, 2026.

Rana Salama, Jason Cai, Michelle Yuan, Anna Currey, Monica Sunkara, Yi Zhang, and Yassine Benajiba. Meminsight: Autonomous memory augmentation for llm agents. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 33124–33140, 2025.

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

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.

Zhanming Shen, Jintao Tong, Shaotian Yan, Chen Shen, Hao Chen, Wentao Ye, Xiaomeng Hu, Rui Miao, Haobo Wang, Junbo Zhao, et al. Purified opsd: On-policy self-distillation without losing how to think. *arXiv preprint arXiv:2607.02234*, 2026a.

Zhennan Shen, Yanshu Li, Qingyu Yin, Chak Tou Leong, Zhilin Wang, Yanxu Chen, Rongduo Han, Sunbowen Lee, and Yi R Fung. On the geometry of on-policy distillation. *arXiv preprint arXiv:2606.07082*, 2026b.

Idan Shenfeld, Mehul Damani, Jonas Hübotter, and Pulkit Agrawal. Self-distillation enables continual learning. *arXiv preprint arXiv:2601.19897*, 2026.

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

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. *arXiv preprint arXiv:2010.03768*, 2020.

Charlie Snell, Dan Klein, and Ruiqi Zhong. Learning by distilling context. *arXiv preprint arXiv:2209.15189*, 2022.

TII Team. The falcon 3 family of open models, December 2024.

Bing Wang, Shaotian Yan, Chen Shen, Sinan Fan, Ximing Li, Rui Miao, Xiaosong Yuan, Zhanming Shen, Jieping Ye, et al. Backtracking when it strays: Mitigating dual exposure biases in llm reasoning distillation. *arXiv preprint arXiv:2605.19433*, 2026a.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.

Hanyang Wang, Lu Wang, Chaoyun Zhang, Tianjun Mao, Si Qin, Qingwei Lin, Saravan Rajmohan, and Dongmei Zhang. Text2grad: Reinforcement learning from natural language feedback. In *International Conference on Learning Representations*, volume 2026, pp. 50627–50664, 2026b.

Shengjie Wang, Guanghe Li, Zonghan Yang, and Yang Gao. Hindsight hint distillation: Scaffolded reasoning for swe agents from cot-free answers. *arXiv preprint arXiv:2605.11556*, 2026c.

Yinjie Wang, Xuyang Chen, Xiaolong Jin, Mengdi Wang, and Ling Yang. Openclaw-rl: Train any agent simply by talking. *arXiv preprint arXiv:2603.10165*, 2026d.

Zhitong Wang, Songze Li, Hao Peng, Shuzheng Si, Yi Wang, Maosong Sun, and Juanzi Li. Envrl: Learn from environment dynamics in agentic reinforcement learning. *arXiv preprint arXiv:2606.17680*, 2026e.

Zongqi Wang, Rui Wang, Yuchuan Wu, Yiyao Yu, Pinyi Zhang, Shaoning Sun, Yujiu Yang, and Yongbin Li. Reward modeling from natural language human feedback. *arXiv preprint arXiv:2601.07349*, 2026f.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. *arXiv preprint arXiv:2409.07429*, 2024.

Jinyang Wu, Shuo Yang, Zhengxi Lu, Fan Zhang, Yuhao Shen, Lang Feng, Haoran Luo, Zheng Lian, Shuai Zhang, Zhengqi Wen, et al. Seed: Self-evolving on-policy distillation for agentic reinforcement learning. *arXiv preprint arXiv:2607.14777*, 2026.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025a. URL https://arxiv.org/abs/2505.09388.

Cheng Yang, Xuemeng Yang, Licheng Wen, Daocheng Fu, Jianbiao Mei, Rong Wu, Pinlong Cai, Yufan Shen, Nianchen Deng, Botian Shi, et al. Learning on the job: An experience-driven self-evolving agent for long-horizon tasks. *arXiv preprint arXiv:2510.08002*, 2025b.

Yongkang Yang, Zhezheng Hao, Hong Zhang, Yi Liu, Xiankun Lin, Wence Ji, Fanjunduo Wei, Jiarui Yu, Qiang Lin, Xiaoyun Liang, et al. Matching supervision to the student's learning capacity: A unified framework for on-policy self-distillation. *arXiv preprint arXiv:2608.08176*, 2026.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35:20744–20757, 2022.

Tianzhu Ye, Li Dong, Qingxiu Dong, Xun Wu, Shaohan Huang, and Furu Wei. Online experiential learning for language models. *arXiv preprint arXiv:2603.16856*, 2026a.

Tianzhu Ye, Li Dong, Xun Wu, Shaohan Huang, and Furu Wei. On-policy context distillation for language models. *arXiv preprint arXiv:2602.12275*, 2026b.

Woongyeng Yeo, Yumin Choi, Taekyung Ki, and Sung Ju Hwang. Hint-sd: Targeted hindsight self-distillation for long-horizon agents. *arXiv preprint arXiv:2605.17873*, 2026.

Guibin Zhang, Jiayang Lyu, Ran Sun, Xinlei Yu, Haoyu Zhao, Qibing Ren, and Shuicheng Yan. Latent on-policy self-distillation. *arXiv preprint arXiv:2608.13040*, 2026a.

Xiaoying Zhang, Yipeng Zhang, Hao Sun, Kaituo Feng, Chaochao Lu, Chao Yang, and Helen Meng. Critique-grpo: Advancing llm reasoning with natural language and numerical feedback. *arXiv preprint arXiv:2506.03106*, 2025.

Yuwei Zhang, Wenhao Yu, Shangbin Feng, Yifan Zhu, Letian Peng, Jayanth Srinivasa, Gaowen Liu, and Jingbo Shang. Bidirectional lms are better knowledge memorizers? a benchmark for real-world knowledge injection. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 16964–16980, 2026b.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: Llm agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 19632–19642, 2024.

Siyan Zhao, Zhihui Xie, Mengchen Liu, Jing Huang, Guan Pang, Feiyu Chen, and Aditya Grover. Self-distilled reasoner: On-policy self-distillation for large language models. *arXiv preprint arXiv:2601.18734*, 2026a.

Ziqi Zhao, Xinyu Ma, Liu Yang, Yujie Feng, Daiting Shi, Jingzhou He, Xin Xin, Zhaochun Ren, and Xiao-Ming Wu. Rosd: Reflective on-policy self-distillation for language model reasoning across domains. *arXiv preprint arXiv:2605.28014*, 2026b.

Jiayi Zhou, Jiaming Ji, Josef Dai, and Yaodong Yang. Sequence to sequence reward modeling: Improving rlhf by language feedback. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 27765–27773, 2025.

Siqi Zhu, Xuyan Ye, Hongyu Lu, Weiye Shi, and Ge Liu. The many faces of on-policy distillation: Pitfalls, mechanisms, and fixes. *arXiv preprint arXiv:2605.11182*, 2026.

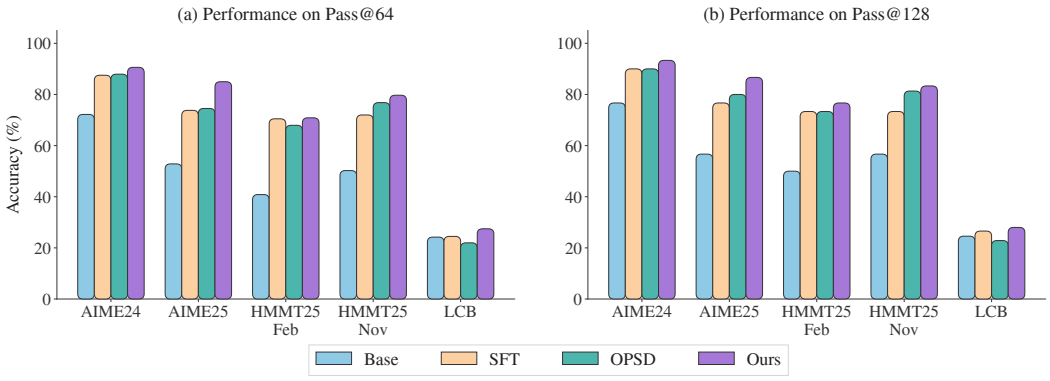



Figure 10: Multi-sample success on mathematics and code: pass@64 (left) and pass@128 (right).

# A Details of Empirical Study

## A.1 Details of the Initial-Accuracy Study

For each model, we construct a single-turn interaction experience for each OpenThoughts problem. Specifically, we sample a student output $a_1$, use the reference solution as feedback $f_1$, and form the interaction trajectory $\tau = (a_1, f_1)$. We then use the same model as the experience extractor $\mathcal{E}$ to obtain the privileged context $c \sim \mathcal{E}(\cdot \mid x, \tau)$. For each problem, we estimate the student's initial accuracy using 128 independent rollouts and assign it to one of six bins: $\{0\}$, $(0, 20]$, $(20, 40]$, $(40, 60]$, $(60, 80]$, and $(80, 100]$. The three models are grouped independently according to their own initial accuracy, and we sample 100 problems from each bin for training.

For each model and accuracy bin, we train OPCD and TRD for 5 epochs as the on-policy and off-policy context-distillation methods, respectively. Both methods are trained on the same sampled problems and their corresponding privileged context. After training, we evaluate the distilled student without privileged context on the same problems using avg@128. All other training and evaluation settings follow those used in our main experiments.

## A.2 Details of the Controlled Knowledge-Injection Study

We construct a synthetic knowledge-injection task to isolate the learning dynamics of context distillation from other factors. Each example follows a single factual template, "*$X$'s $R$ is $Y$*," with the corresponding query "*What is $X$'s $R$?*" Both the subject $X$ and the words composing the answer $Y$ are randomly generated pseudo-words and $R$ is a random relationship. We generate these pseudo-words using a syllable-based grammar and filter out real English words to minimize overlap with the model's prior knowledge. We use a single relation and query template throughout the dataset to avoid variation in relation difficulty. In addition, every pseudo-word appears only once across the entire dataset, so different facts share no subject or answer words. The synthetic fact is directly provided to the privileged teacher as context, while the student receives only the query.

To control the length of the behavior to be learned, we construct answers whose tokenized lengths are exactly $L \in \{4, 8, 16, 32\}$. Specifically, each answer is formed by concatenating pseudo-words and retaining only combinations whose tokenization contains exactly $L$ tokens at the generation position. No padding or rewriting is used. All length groups are generated from the same pseudo-word distribution and are mixed within the same training run, making answer length the primary controlled variable. We conduct this study with Qwen3-4B using the same on-policy and off-policy context distillation procedures as above, training for 200 epochs and evaluating every 5 epochs. Each length group contains eight facts. At evaluation time, the student receives no privileged context and independently generates eight responses for each fact. We report exact-match accuracy averaged first over the eight generations for each fact and then over all facts in the corresponding length group. All other training and evaluation settings follow the main experiments.

# B PARAMETER-UPDATE GEOMETRY

We provide additional details for the parameter-update analysis in Section 5.3. Following the diagnostics of Shen et al. (2026b), we examine how different context-distillation objectives modify the pretrained parameters over training. We consider both the *scale and support* of parameter changes and their effect on the singular structure of the pretrained weights. In addition to mathematical reasoning reported in the main text, we conduct the same analysis on code generation and ALFWorld.

## B.1 ANALYSIS PROTOCOL

All analyses use Qwen3-8B and measure parameter changes relative to the shared initial checkpoint. For each weight matrix $W$, we denote its initialization by $W_0$ and its value at training step $t$ by $W_t$. We analyze checkpoints at $t \in \{20, 40, 60, 80, 100\}$. For ALFWorld, which is trained for more than 100 updates, we use the same 100-step window to enable direct comparison with the reasoning and coding runs.

To cover different model depths while keeping the analysis tractable, we analyze the query, key, value, and output projection matrices from every fourth transformer layer, $i \in \{0, 4, 8, \ldots, 32\}$, yielding 36 matrices in total. Each diagnostic is computed independently for every matrix and then averaged equally across the 36 matrices; thus, larger matrices do not receive additional weight.

Stored weights are converted to FP32 before computing parameter differences. Singular-value decompositions are computed in FP64 for numerical stability. The same matrices, checkpoints, and aggregation procedure are used for all methods and tasks.

## B.2 METRIC DEFINITIONS

We use four complementary diagnostics. Update magnitude and visible update sparsity characterize how much and where the parameters change, while principal angle and normalized spectral shift characterize changes in the singular geometry of the pretrained weights.

**Cumulative update magnitude.** For each matrix, we measure its displacement from initialization as

$$\Delta W_t = W_t - W_0, \qquad d_t(W) = \|\Delta W_t\|_F. \tag{1}$$

This quantity measures the net displacement at step $t$, rather than the sum of successive update norms. We do not normalize it by matrix size or by $\|W_0\|_F$. The reported update scale is the mean of $d_t(W)$ over the analyzed matrices.

**Visible update sparsity.** To distinguish substantial parameter movement from small numerical changes, we mark an entry as visibly changed when

$$M_{t,ij} = \mathbb{I}[|W_{t,ij} - W_{0,ij}| > \eta \max(|W_{0,ij}|, |W_{t,ij}|)], \qquad \eta = 10^{-3}. \tag{2}$$

For a matrix containing $n_W$ entries, we define

$$\mathrm{Sp}_t(W) = 1 - \frac{1}{n_W} \sum_{i,j} M_{t,ij}. \tag{3}$$

Higher values therefore indicate that a larger fraction of coordinates remain within the relative-change threshold. Equivalently, $1 - \mathrm{Sp}_t(W)$ is the visibly changed-coordinate fraction reported in our analysis.

This diagnostic follows the relative-change criterion used in prior geometric analyses of distillation, but our implementation applies the criterion directly to the stored weights after FP32 conversion and does not perform an additional bf16 round trip. We therefore interpret it as *visible update sparsity*, rather than as an indicator of whether a parameter received gradients.

**Principal-subspace rotation.** We next measure how much training rotates the dominant singular subspace. Let

$$W_0 = U_0 \Sigma_0 V_0^\top, \qquad W_t = U_t \Sigma_t V_t^\top. \tag{4}$$

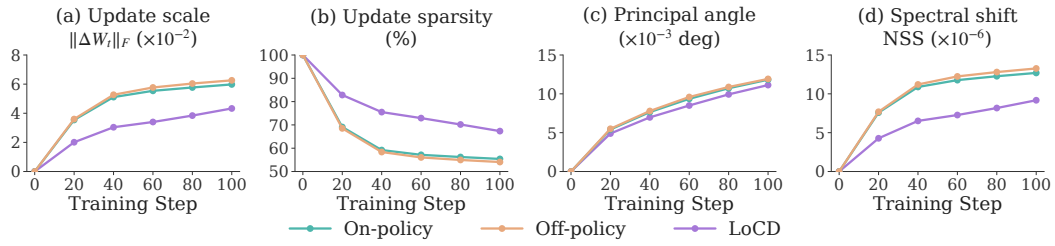



Figure 11: Parameter-update geometry during mathematical-reasoning training.

We retain the leading $k = 512$ left singular vectors, $U_{0,k}$ and $U_{t,k}$. Their principal angles are

$$\theta_{t,i} = \arccos\left(\sigma_i\left(U_{0,k}^\top U_{t,k}\right)\right), \qquad i = 1, \ldots, k. \tag{5}$$

The singular values of the overlap matrix are clipped to $[0, 1]$ before applying $\arccos$. For each weight matrix, we use the median principal angle in degrees and then average this statistic across matrices. Smaller values indicate less rotation of the dominant left singular subspace. This metric measures subspace orientation and should not be interpreted as the rank of the parameter update.

**Normalized spectral shift.** Principal angles characterize changes in singular vectors; we additionally measure changes in the singular values themselves. Let $\sigma(W)$ denote the complete singular-value spectrum of $W$ in descending order. We compute

$$\mathrm{NSS}_t(W) = \frac{\|\sigma(W_t) - \sigma(W_0)\|_2}{\|\sigma(W_0)\|_2}. \tag{6}$$

Unlike the principal-angle diagnostic, NSS uses the full singular spectrum. Lower NSS indicates a smaller relative change in the singular values of the pretrained weight matrix. The reported NSS is averaged equally across the analyzed matrices.

### B.3 Results Across Tasks

Across the three settings, the most consistent distinction appears in update magnitude and visible update sparsity: LoCD stays closer to initialization and changes fewer coordinates than either on-policy or off-policy context distillation. The spectral diagnostics show a similar but somewhat weaker pattern.

**Mathematical reasoning.** Figure 11 reproduces the main-text analysis. LoCD exhibits smaller cumulative displacement throughout training. At step 100, its update magnitude is approximately 28–31% smaller than those of the on-policy and off-policy baselines. The visibly changed-coordinate fraction is 32.66%, compared with 44.60% for on-policy and 45.93% for off-policy distillation. LoCD also yields a lower NSS ($9.17 \times 10^{-6}$ versus $12.69 \times 10^{-6}$ and $13.27 \times 10^{-6}$). The principal-angle difference is comparatively modest ($0.01113°$ versus $0.01182°$ and $0.01192°$), indicating that the clearest separation lies in update magnitude and visible support rather than a substantially different rotation of the dominant subspace.

**Code generation.** Figure 12 shows a closely matching pattern on code generation. At step 100, the mean update magnitude is $0.0432$ for LoCD, compared with $0.0577$ for on-policy and $0.0632$ for off-policy distillation. The corresponding visibly changed-coordinate fractions are 32.93%, 42.74%, and 45.80%, respectively. LoCD also exhibits a smaller spectral shift. Principal-angle differences are less pronounced and do not follow a strict ordering at every checkpoint, again suggesting that the main distinction is in the scale and support of the parameter update.

**ALFWorld.** The separation is even larger on ALFWorld (Figure 13), particularly relative to off-policy distillation. At step 100, the mean update magnitude is $0.0381$ for LoCD, $0.0537$ for on-policy distillation, and $0.1046$ for off-policy distillation, making the LoCD displacement 63.6% smaller than the off-policy baseline. The visibly changed-coordinate fractions are 29.70%, 38.75%, and 59.20%, respectively. LoCD also exhibits smaller principal-subspace rotation and NSS throughout the analyzed window.

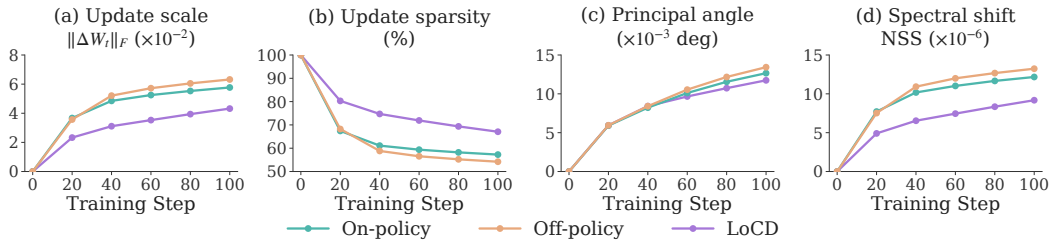



Figure 12: Parameter-update geometry during code-generation training.

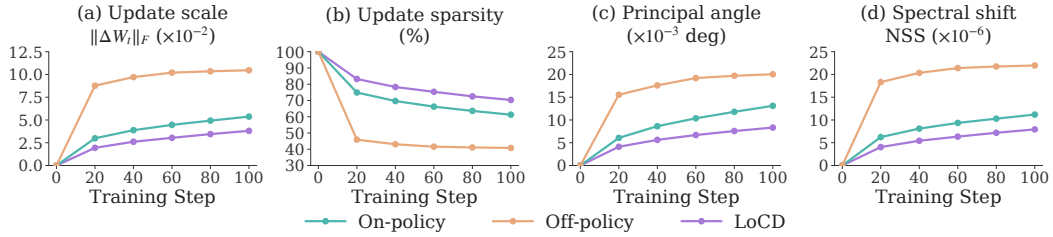



Figure 13: Parameter-update geometry during ALFWorld training.

**Overall pattern.** Across mathematical reasoning, code generation, and ALFWorld, LoCD consistently induces smaller cumulative displacement and a smaller fraction of visibly changed coordinates in the analyzed attention projections. Spectral changes are also generally smaller, while differences in dominant subspace rotation are more modest on reasoning and coding. Together with the task-performance results, these analyses show that LoCD achieves stronger performance while modifying the pretrained parameters more locally.

These diagnostics characterize the parameter changes observed in the analyzed attention projections. They do not imply fewer trainable parameters, lower optimization cost, low-rank updates, or that smaller parameter changes are themselves the cause of LoCD's performance gains.

# C Prompt Templates

This section provides the prompts used in our math reasoning experiments. Text enclosed in curly braces denotes placeholders that are dynamically replaced with the corresponding instance-specific content at runtime.

## C.1 Overall Assessment

We use the following prompt to obtain an overall assessment of a student's solution by comparing it with the corresponding reference solution. The assessment summarizes the student's overall approach, identifies where its reasoning begins to diverge from the reference solution, characterizes the nature of the divergence, and suggests the direction of correction.

**Overall-Assessment Prompt**

```
You are a rigorous mathematics problem-solving coach. You will be
    given a
problem, a student's attempted solution, and the reference solution.
    Your task
is to compare the student's attempt with the reference solution and
    provide a
single overall assessment of the student's attempt.
```

```
# Problem
{question}

# Student Attempt
{response}

# Reference Solution
{solution}

# Task
Compare the student's attempt against the reference solution and provide a
single overall assessment. Your assessment must cover the following four
points, in order:

1. What overall approach or method the student uses.

2. At which stage the student's reasoning first departs from the line of
reasoning in the reference solution. Describe the substance of the divergence;
you do not need to quote or pinpoint the exact sentence.

3. The nature of the divergence: conceptual (a different choice of method or
solution route), execution-related (the same general route, but an error in
derivation or calculation), or interpretational (the problem conditions are
understood differently from the reference solution).

4. From the perspective of the reference solution, how the student's approach
should be adjusted next. Describe only the direction of the correction.

Assessment requirements:
- Be specific to the current problem. Explicitly mention the relevant
  quantities, relationships, equations, or variable names; avoid vague
  statements such as "this part differs from the reference."
- Evaluate only the mathematical problem-solving content. Do not comment on
  formatting, wording, verbosity, or presentation.
- Do not locate or enumerate errors sentence by sentence; that is handled in a
  separate stage.
- If the student's attempt is fully consistent with the reference solution,
  simply state that.

# Output Format
Return strict JSON and no additional text:
{"overall_assessment": "<one coherent paragraph organized according to the four
points above and specific to the quantities and relationships in the current
problem>"}
```