

000 001 002 003 004 005 006 007 008 009 010 PATCHING GAPS IN LLM REASONING WITH INTERVENTIONAL TRAINING

005 **Anonymous authors**

006 Paper under double-blind review

009 ABSTRACT

011 Reinforcement learning (RL) training of large language models (LLMs) is limited
 012 by the policy’s ability to generate rollouts with non-zero rewards: without such
 013 rewards, the policy is not updated and learning is stalled on hard problems, which
 014 are problems that the policy consistently fails to sample any correct rollouts for.
 015 We find that many hard problems remain unsolved due to the repeated generation
 016 of incorrect intermediate steps in a long reasoning trace; identifying and fixing
 017 these requires performing better *credit assignment*. But existing approaches for
 018 credit assignment are either impractical or impose a substantial data-writing burden
 019 on oracles (e.g., humans). In this paper, we introduce **Interventional Training**
 020 (InT), a framework that leverages single-step oracle interventions to improve LLM
 021 reasoning. Given a reasoning attempt and ground-truth answer, the oracle detects
 022 and then provides language feedback on a single intermediate reasoning step,
 023 which is much cheaper than obtaining a full reasoning trace. InT then *patches*
 024 the LLM by running supervised fine-tuning on the on-policy rollout up to the
 025 error, followed by the correction from the oracle. RL on this patched model now
 026 generates counterfactual traces and with merely ≈ 100 interventions from the oracle,
 027 InT solves 16% more hard test problems that were previously unsolved (only zero
 028 rewards) and also improves performance across multiple standard evals.

029 1 INTRODUCTION

030 Post-training large language models (LLMs) with reinforcement learning (RL) has proven to be a
 031 highly effective strategy for improving their reasoning capabilities. In a typical RL recipe, we update
 032 the current policy on a given problem by first sampling from the policy multiple rollouts conditioned
 033 on the problem, and then positively reinforcing the policy on correct rollouts while down-weighting
 034 the likelihood of incorrect ones (DeepSeek-AI et al., 2025). However, if no rollout ends up being
 035 correct, all traces receive zero reward. This poses a barrier to scaling RL to harder problems where
 036 the policy fails to acquire any non-zero reward. When faced with such a scenario, a practitioner
 037 could modify pre-training or mid-training run typically before RL to retrain a base model capable of
 038 attaining reward, but the effect of these procedures on subsequent RL is still not well understood.

039 What do the reasoning traces on such problems unsolved during training look like? While several such
 040 problems lie far outside the scope of the pre-trained model’s capabilities, we find that a substantial
 041 (**near 30%**) chunk of problems that we cannot train on today are those where the model makes an
 042 *execution* mistake in its trace, which derails subsequent reasoning, and results in an incorrect answer
 043 even with several parallel attempts. Such mistakes are more likely to occur at long lengths that are
 044 composed of many steps. This raises a key question: *how can we still train models on hard problems*
 045 *on which they fail to sample correct traces largely due to execution errors?*

046 In principle, addressing this challenge requires addressing the problem of *credit assignment*: if we
 047 could pinpoint the intermediate step at which a reasoning trace goes astray, and feed in a more accurate
 048 alternate step, we could train the model to correct its course from that point onward. However, credit
 049 assignment in long reasoning traces is notoriously difficult. One way of identifying an incorrect
 050 step is to sample multiple rollouts from each prefix in the long trace, and find the step that drops the
 051 probability of success, but this can be a prohibitively high sampling cost for long length reasoning
 052 models (Kazemnejad et al., 2024). Another option is to train process reward models (PRMs) (Setlur
 053 et al., 2024b; Lightman et al., 2023b) on oracle data and use them to directly evaluate intermediate
 reasoning steps. Training accurate PRMs often is quite sample inefficient (Luo et al., 2024), and can

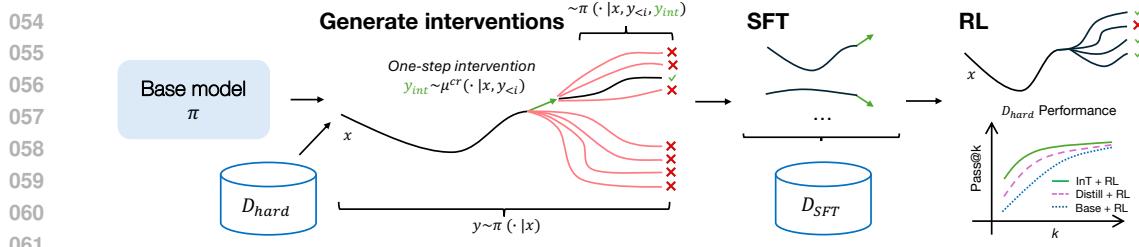


Figure 1: **Intervention training for patching LLMs during RL.** Instead of relying on full expert reasoning traces or attempting to find rare correct rollouts during RL, InT introduces *single-step*, oracle interventions that “patch” incorrect intermediate steps in model-generated reasoning traces. Conditioned on these localized corrections, the model can generate counterfactual continuations that succeed where the original failed. We then distill these interventions into the model via supervised fine-tuning before resuming RL, enabling effective credit assignment and continued progress even on problems that were previously unsolvable with standard RL.

be particularly hard for long reasoning traces (Kim et al., 2025). As a result, existing approaches to credit assignment remain impractical for long reasoning traces. That said, we can improve over these approaches if we assume access to an oracle (e.g., a human) that provides some kind of intermediate feedback on *just* the incorrect step, and if the number of calls to such an oracle is forced to be limited.

Motivated by this, we introduce **Interventional Training (InT)** (Figure 1), an approach for effective credit assignment in long reasoning traces from a model that has plateaued during RL. Here, instead of requesting complete and long expert reasoning traces, InT solicits *single-step interventions* from an oracle (e.g., a human, another LLM, or a specialized verifier). Given a model-generated reasoning attempt and the ground-truth answer, the oracle replaces exactly one critical and incorrect intermediate step with a corrected version (e.g., fixing an incorrect approximation in a long math answer, see Fig. 2). Conditioned on this intervention, the model can then generate alternate counterfactual traces that may succeed where the original failed. In this way, interventions provide a low-cost mechanism to discover correct reasoning traces. Next, InT internalizes these interventions into the model weights by running supervised fine-tuning on the interventions, and then continues the RL run that had previously plateaued. By “patching” the model on single-step interventions in this manner, InT makes it possible for the model to attain non-zero reward signals even on otherwise unsolvable problems, enabling effective training on problems that are inaccessible with RL.

Since we are using the oracle for localized credit assignment, why not use the oracle to generate entire reasoning traces on the hard problem that we can clone? Requesting localized interventions requires far less data-writing effort than producing full ground-truth solutions in the model’s output format, and can flexibly incorporate feedback from humans, specialized tools that an LLM interacts with, or other oracle models. Moreover, we find that even if we had full length reasoning traces, since they typically lie outside our base model’s distribution, cloning them leads to memorization and completely distorts the base model’s next-token distribution.

By running RL on a set of 64 training problems, InT achieves a 6.09% (3.12% to 9.03%) gain in pass@64 and solves 14% more problems on a challenging held out test set. In contrast, simply distilling the oracle or running SFT on oracle data achieves only a 3.51% gain in pass@64. On standard reasoning benchmarks, InT leads to an average improvement of 1.92% across 7 standard math reasoning benchmarks, showing that patching the model with InT does not degrade existing model capabilities. These results show that InT offers a simple, deterministic way to patch model behavior, improving performance on new problems while preserving or improving existing reasoning capabilities.

2 RELATED WORK

Credit assignment in LLM reasoning. The effectiveness of long length RL with outcome rewards (DeepSeek-AI et al., 2025) is often crippled by credit assignment: it is unclear which intermediate steps in a long response should be “credited” for the outcome reward. While one might surmise that sampling enough rollouts should address this problem, note that the difficulty of credit assignment also greatly increases with the horizon (Setlur et al., 2025a). While most methods reward each token with the outcome level advantage (Yu et al., 2025), others use process reward models (PRMs) to assign dense token or step-level rewards (Lightman et al., 2023a; Wang et al., 2024; Qu et al., 2025b) that can reinforce correct steps and promote unlearning of incorrect ones. Although PRMs may

108 improve RL compute efficiency (Setlur et al., 2024b;a), we often require costly rollouts (Luo et al.,
109 2024; Kazemnejad et al., 2024) to reliably estimate these reward signals. We instead leverage oracles
110 to detect individual mistakes, bypassing the compute required to train entirely new PRMs or to
111 perform full credit assignment on the full reasoning trace; even when the oracle is itself a reasoning
112 model, it is given the ground-truth response and only needs to perform a comparative analysis to
113 identify an intervention. While our method shares the general idea of using generative models as
114 verifiers (Zhang et al., 2024; Liu et al., 2025; Kim et al., 2025; Khalifa et al., 2025), it is distinct in
115 that we task the oracle with explicitly pointing out the location of a single, key mistake, rather than
116 verifying every step and judging the solution. Finally, we use the outputs of the oracle to improve RL
117 training rather than inference-time methods (e.g., best-of- N search), which prior works focus on.
118

119 **Learning from natural language feedback.** Another related line of work explores utilizing natural
120 language feedback to improve RL training. Such works typically leverage the feedback to refine
121 rollouts that are then used to improve the policy. Chen et al. (2024) combine human feedback and
122 a separate refinement model to improve policy-generated outputs that are then distilled back into
123 the policy via SFT. Yan et al. (2025) use a teacher model to generate correction trajectories for off-
124 policy RL, while Zhang et al. (2025b) conduct critique-guided self-refinement to generate correction
125 trajectories, again for use in off-policy RL. Unlike these works, our work considers generating short,
126 targeted natural language feedback to correct individual steps within what are otherwise purely
127 on-policy trajectories. As we discuss later on, this allows us to achieve substantial improvements
128 without making significant changes to the standard RL training recipe.
129

130 **Intervention training outside of LLMs.** Applying interventions at exact points of failure has been
131 explored in domains outside of LLM training, for example, in dexterous manipulation (Hu et al.,
132 2025) and imitation learning (Kelly et al., 2019). The class of intervention methods across these
133 different domains generally shows improved data efficiency and faster convergence of reward curves
134 compared against naive behavior cloning method due to superior credit assignment. In our work, we
135 examine whether applying such ideas to language model RL training can reap similar benefits.
136

3 PRELIMINARIES AND PROBLEM STATEMENT

137 Reasoning LLMs still struggle to
138 solve certain training problems. One
139 might expect that training on hard
140 problems during RL helps improve
141 the model’s success rate on analo-
142 gous hard problems at test time, which
143 makes it important to be able to derive
144 learning signal on some such prob-
145 lems. To better understand the fail-
146 ure modes of LLMs on difficult rea-
147 soning questions, we inspect rollouts
148 from the reasoning LLM Qwen3-4B-
149 Instruct and find that a substantial
150 number of model failures stem from a
151 failure of *execution* and an inability to
152 *recover* from failure: even though the
153 model often takes a correct high-level
154 approach, it gets derailed during an in-
155 termediate step when attempting to ex-
156 ecute on this high-level approach. We
157 demonstrate some examples of these
158 failures in Figure 2 and Appendix B.
159 As shown in Figure 4 and as noted
160 by Sinha et al. (2025), failure in ex-
161 ecution (vs. high-level strategy) is a
162 common occurrence in long reasoning
163 traces, especially later in the trajectory.
164 Our objective is to improve model per-
165 formance on challenging reasoning tasks
166 in which such execution failures result in
167 scarce positive rewards for RL.

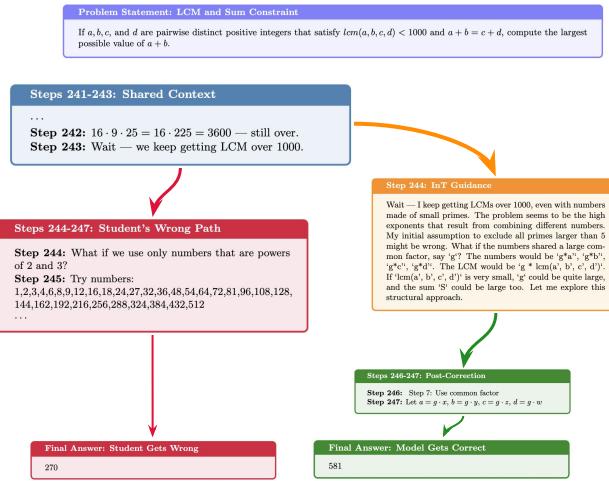


Figure 2: **Example execution error of Qwen3-4B-Instruct.** Note that the model deduces an incorrect conclusion from going over 1000 and continues trying small primes, which an oracle intervention (via our approach, InT) addresses. We find that 40.6% stem from execution errors. This showcases the potential of fixing execution errors directly for RL, without needing to rerun pre-training or mid-training.

162
 163
 164
 165
 166
 167
 168
Notation. To build our approach, we first define some relevant notation. LLM training for reasoning
 typically involves an LLM π , a binary reward $r(\mathbf{x}, \mathbf{y}) \in \{0, 1\}$ for inputs $\mathbf{x} \sim \rho$, and model outputs
 $\mathbf{y} \sim \pi(\cdot | \mathbf{x})$. Common paradigms for training LLMs include supervised fine-tuning (SFT) and
 online policy gradient methods (henceforth referred to as RL). SFT fine-tunes a model by maximizing
 likelihood on a dataset (ideally consisting of correct traces \mathbf{y} for problems \mathbf{x}). RL, in contrast, samples
 candidate rollouts from its own policy distribution (i.e., on-policy) and trains to maximize the reward
 of these rollouts. In its simplest form, RL training for LLMs uses the policy gradient:

$$\theta' \leftarrow \theta + \alpha \cdot \mathbb{E}_{\mathbf{y} \sim \tilde{\pi}(\cdot | \mathbf{x})} [r(\mathbf{x}, \mathbf{y}) \cdot \nabla_{\theta} \log \pi_{\theta}(\mathbf{y} | \mathbf{x})], \quad (1)$$

171
 172
 173
 where $\tilde{\pi}$ is the policy used to generate samples that go in for training. In RL methods such as GRPO,
 $\tilde{\pi} = \pi_{\text{old}}$ is a periodically updated copy of π , and the reward $r(\mathbf{x}, \mathbf{y})$ is normalized by subtracting a
 baseline to form the advantage, which serves as the multiplier to $\nabla_{\pi} \log \pi$ instead of r :

$$A_i(\mathbf{x}, \mathbf{y}_i) = r(\mathbf{x}, \mathbf{y}_i) - 1/n \sum_i \bar{r}(\mathbf{x}, \mathbf{y}_i).$$

174
 175
 176
Credit assignment. As discussed above, a substan-
 177 tial fraction of errors made by models on the training
 178 distribution correspond to execution errors. This nat-
 179 urally means that while some tokens/steps in a model
 180 response are on the right track towards solving the
 181 problem at hand, a different subset of tokens derails
 182 the model onto not attaining the right final answer. If
 183 we could identify the identity of these tokens/steps
 184 and the extent of their influence (i.e., “credit”) in
 185 affecting the correctness of the final answer, then
 186 this problem could be solved as long as the model is
 187 able to find alternative steps that do actually succeed.
 188 This process is called **credit assignment** (Setlur et al.,
 189 2024a). RL algorithms rely on self-generated rollouts
 190 to guide the process of credit assignment.

191 However, performing credit assignment solely from
 192 outcome-level rewards is highly challenging. When the advantage is positive, every token probability
 $\pi(\mathbf{y}_i | \mathbf{x}, \mathbf{y}_{<i})$ across the sequence is equally reinforced; when it is negative, all tokens are equally
 193 discouraged. Over long reasoning traces, such uniform updates are not effective at performing credit
 194 assignment: tokens that played no role in reaching the solution may still be upweighted (“higher
 195 credit”), while tokens that were correct but followed by later mistakes may be suppressed (“lower
 196 credit”). Even when we know which tokens failed at a particular problem, we need to search for other
 197 steps to pursue to get to a correct answer. On hard problems, this noise can overwhelm the signal,
 198 hindering the model from making progress. Correctly assigning credit is therefore crucial.

Problem Statement

201 A significant fraction (40.6%) of mistakes made by LLMs on hard problems stem from
 202 execution errors. We aim to address these errors by performing better credit assignment
 203 (reinforcing correct steps, unlearning incorrect steps, and finding alternative completions).

4 INT: INTERVENTIONAL TRAINING FOR CREDIT ASSIGNMENT

206 Given a problem, when a reasoning LLM repeatedly falters, one of the most direct approaches of
 207 “patching” it on this problem is to behavior clone oracle (e.g., human) generated off-policy solutions
 208 to this problem. However, oracle full-length solutions to problems are highly off-policy with respect
 209 to the policy itself. Prior works argue (Zhang et al., 2025a; Setlur et al., 2025a; Kang et al., 2024), and
 210 we further show, that distilling such data interferes with the model’s reasoning capabilities, especially
 211 on out-of-distribution problems. *Is there a simpler, more data-efficient, and effective way to leverage
 212 oracles for improving credit assignment?* In this section, we develop our approach for doing so.

213 Our key idea, **Int** (interventional training), is to structure oracle feedback as single-step corrective
 214 interventions on the base model’s (model plateaued during RL training) output reasoning trace: each
 215 intervention provides short natural language feedback at an intermediate step proposed by the base
 model, guiding the subsequent completion, particularly in ways different from simply continuing the

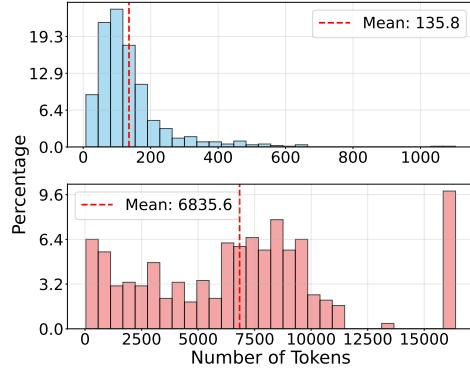


Figure 3: **Interventions are short.** Top: Lengths of interventions typically span under 200 tokens, while full solutions are much longer (bottom).

rollout (see illustration in Fig 1). These interventions are easy to obtain and avoid the type mismatch problem described above since they are simply ~ 100 tokens long (Fig. 3). Our experiments show that interventions provide a sample-efficient mechanism for incorporating oracle feedback to do credit assignment. With ≤ 80 interventions, we improve accuracy on tasks where the base model previously obtained no reward, and also advance state-of-the-art across various reasoning benchmarks.

4.1 INT INTERACTION PROTOCOL

To instantiate **IntT**, we collect data that pinpoints where a model’s reasoning trace first goes wrong and provides a corrective step at that location. In principle, the corrective step of the intervention comes from an oracle, which we simulate using a larger, proprietary reasoning model denoted by μ^{cr} . This oracle does not need to be capable of solving the problem by itself, but it should be capable of comparing the base LLM π ’s solution trace y , with the ground truth solution y^* on problem x . Assuming that y can be segmented into k_0 reasoning steps (based on simple keywords like “wait”, “maybe”, or “\n” based segmentation), μ^{cr} identifies the first step $i \in [k_0]$ where an error occurs that itself is not corrected by the base LLM, and outputs a corrective intervention $y_{\text{int},i}$. This intervention may either prevent the error from arising at step i or repair it immediately afterward. This approach requires the oracle $\mu^{\text{cr}}(\cdot | y^*, y, x)$ to simply perform step-by-step comparison operations until the *first* index i where the responses are incorrect, and then return a short intervention.

An example of this interaction protocol between an LLM and the oracle is shown in Figure 2. We implement this idea with small 1.7B and 4B reasoning LLMs as our base models, and Gemini 2.5 Pro as the oracle, and note that most interventions occur mid-trace rather than at the beginning. The position of interventions also demonstrates that **IntT** is fundamentally different from “hint-conditioning” approaches (Qu et al., 2025a), which prepend a “hint” before reasoning begins.

Empirically we find that continuing rollouts from just before the identified error location $\pi(\cdot | x, y_{<i})$ significantly improves the likelihood of reaching a correct solution compared to rollouts from the start of the problem x . Appending the corrective step from the oracle to the base model-generated steps before the error and then continuing to sample from the base model $\pi(\cdot | x, y_{<i}, y_{\text{int},i})$ yields further gains, confirming that both localizing mistakes and injecting short corrective interventions are individually useful. Some concrete examples of interventions are shown in Figure 2, with additional cases in Appendix B. We report the aggregate performance and number of problems solved by conditioning on interventions in Table 1.

4.2 PATCHING THE BASE LLM WITH ORACLE INTERVENTIONS

Having established the intervention protocol, the next step is to “patch” the gaps in the base LLM’s reasoning capabilities using these interventions. A natural approach is to apply supervised fine-tuning (SFT) on the collected intervention data. Since nearly all tokens in the correct partial rollout $(y_{<i}, y_{\text{int},i})$ originate from the base LLM $y_{<i}$, with the intervention step $y_{\text{int},i}$ as the only exception, we simply need to teach the model how to internalize and sample a similar single-step intervention when running on its own without access to an oracle. In principle, this can be done by fine-tuning the base LLM on the partial solution $((x, y_{<i}))$ as input, and the intervention $y_{\text{int},i}$ as output. However, as shown in prior work (Qu et al., 2024; Kumar et al., 2024), cloning *only* the tokens present in intermediate steps conditioned on a self-generated

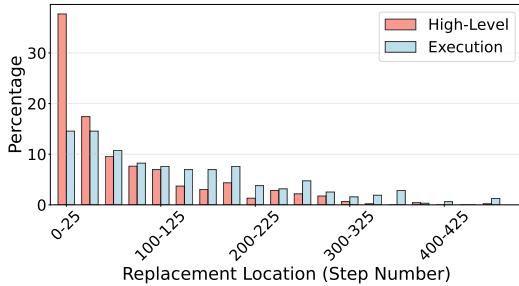


Figure 4: *Histogram of intervention locations by error category*. High-Level errors occur early on in the trace but execution errors persist throughout model rollouts. The position of interventions also demonstrates that **IntT** is fundamentally different from “hint-conditioning” approaches (Qu et al., 2025a), which prepend a “hint” before reasoning begins.

Configuration	Nonzero Acc.	Accuracy
Naive	98/235	0.97%
From intervention	120/235	2.37%
With intervention	145/235	3.72%
Prefix, no suffix, filter	202/235	7.71%
+ no filter	196/235	5.06%
+ no prefix	162/235	2.87%
+ suffix	111/235	2.31%

Table 1: **Intervention-augmented configurations.** From the base model “Naïve” rollout; “from intervention”: from error step identified by oracle; “with intervention”: rollout with one-step oracle guidance; during SFT “prefix”: clone $y_{<i}$; “no suffix”: do not clone; “filter”: keep only interventions yielding correct rollouts from the base policy, when conditioned on them.

270 prefix is often insufficient: the model may generate an alternate prefix $y'_{<i}$ in its rollout on the
 271 very same problem x , which might get derailed for a different reason resulting still in close to zero
 272 successes on this hard problem. Theoretically, this issue can be fixed if our training data consisted of
 273 several prefixes drawn from the base LLM paired with corresponding oracle interventions, but doing
 274 so will degrade the sample efficiency of our approach significantly, and hence is not desirable.

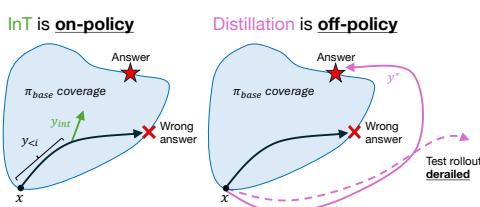
275 Therefore, following the recommendations of Qu et al. (2024), we choose to clone *both* the initial
 276 prefix $y_{<i}$ sampled from the base LLM itself and the intervention $y_{int,i}$, even if the prefix itself is
 277 suboptimal (in fact, we do not even separately evaluate the quality of the prefix). We also experimented
 278 with alternative strategies, such as cloning only the intervention conditioned on the prefix, or cloning
 279 the entire trace (base prefix, oracle intervention, and successful completion from the base LLM
 280 conditioned on the intervention), but found these to perform worse. In the latter approach, cloning
 281 the successful completion significantly and unnecessarily reduces entropy and diversity of model
 282 generations, which hurts on-policy exploration in the subsequent RL training that we do.

283 Concretely, as we show in Table 1, running SFT on both the prefix $y_{<i}$ and the oracle intervention
 284 $y_{int,i}$ for any given problem x performs best. In particular, we only SFT on “filtered” problems, *i.e.*,
 285 only cloning interventions on problems that lead to successful traces from the base policy, when
 286 conditioned on the intervention. We find that adding this filter, and not cloning the completion
 287 (conditioned on the intervention) is helpful mainly to retain the diversity and entropy of the base
 288 model, and to alter its next-token distribution only on states where doing the SFT significantly
 289 improves the accuracy on hard problems. This “patched” model produced by **Int** now serves as a
 290 good initialization for continuing the RL run that had previously plateaued to improve performance.

291 4.3 CONTINUING REINFORCEMENT LEARNING POST-TRAINING

293 After fine-tuning the base LLM on intervention
 294 data for a few steps, we continue to RL post-
 295 training, re-initializing from the patched model.
 296 If the model has successfully internalized the in-
 297 tervention, we expect at least some on-policy roll-
 298 outs to attain non-zero reward on problems the
 299 unpatched/base model could not solve. Once this
 300 occurs, RL training from patched model can rein-
 301 force corrective behaviors while suppressing seg-
 302 ments that do not lead to success. As a result,
 303 it can now extract learning signal from problems
 304 that previously provided none, leading to improve-
 305 ments in both training and test performance. In
 306 contrast, as we will show, continuing to run naïve
 307 RL on the unpatched model would continue sharp-
 308 ening the model’s distribution on problems it can
 309 solve correctly but not to 100% accuracy, and do-
 310 ing so, reduces model diversity and cripples it from
 311 solving these problems.

312 5 WHY ARE INTERVENTIONS EFFECTIVE?



320 **Figure 5: InT improves over distillation.** By
 321 cloning mostly on-policy rollouts with minimal or-
 322 acle edits, InT preserves base model skills while still
 323 patching errors, avoiding the distribution shift that
 324 harms reasoning in full-trace cloning.

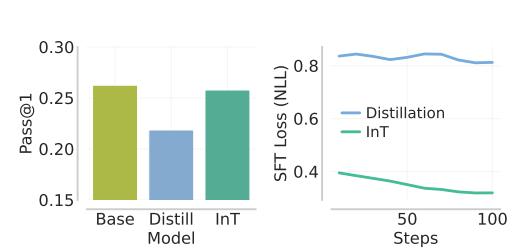
Algorithm 1 InT: Intervention Training

Require: Base LLM π , Oracle μ^{cr} , Problems $\{(x, y^*)\}$

- Data Collection:** $\mathcal{D}_{InT} \leftarrow \{\}$
- for** each x, y^* **do**
- Generate $y \sim \pi(\cdot|x)$; segment into steps
- $i \leftarrow \mu^{cr}(y^*, y, x)$ (first error)
- if** $i \neq \emptyset$ **then**
- $y_{int,i} \sim \mu^{cr}(\cdot|y^*, y, x)$
- $\tilde{y} \leftarrow [y_{<i}, y_{int,i}]$
- $\mathcal{D}_{InT} \leftarrow \mathcal{D}_{InT} \cup (x, \tilde{y})$
- end if**
- end for**
- Patching:** $\pi' \leftarrow SFT(\pi, \mathcal{D}_{InT})$
- RL training:** $\pi'' \leftarrow RL(\pi', \{x\}, \{y^*\})$
- return** π''

312 solving these problems.

313 **Figure 6: SFT on oracle traces reduces test perfor-
 314 mance. InT meanwhile retains the base model per-
 315 formance, thereby providing a good initialization for RL.**



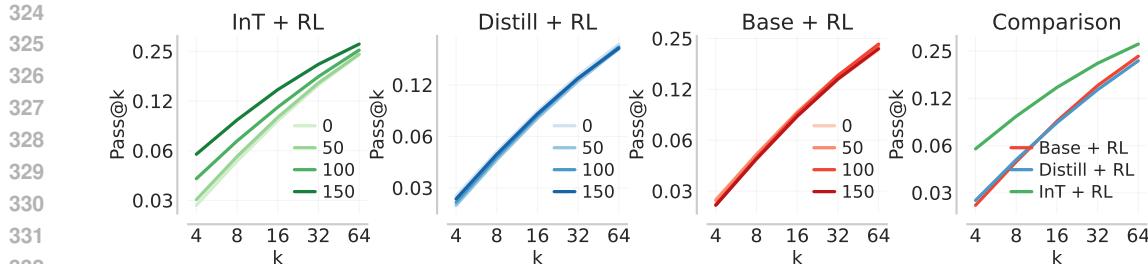


Figure 7: **Pass@ k across RL training iterations:** We plot pass@ k performance from 0 to 150 RL iterations for three initializations: (i) base model patched with InT; (ii) base model distilled on oracle traces; and (iii) directly the base model. Int patched model improves pass@ k consistently while others mainly sharpen.

The most direct way to **distill** oracle information $\mathbf{y}^* \sim \mu^{\text{cr}}(\cdot | \mathbf{x})$ from an oracle μ^{cr} is to perform behavior cloning, i.e., increasing $\log \pi_\theta(\mathbf{y}^* | \mathbf{x})$. To investigate the effectiveness of this approach, we conduct SFT on a set of oracle-generated (Gemini 2.5 Pro) rollouts, and find that this severely impairs the reasoning ability of the resulting model, as illustrated in Figure 6(b). In contrast, conducting SFT maintains the base model performance, thereby providing a better initialization for downstream RL.

Why is excessive deviation from the base model problematic? When the base model is already competent on some tasks/problems, attempting to “patch” its behavior on the subset of unsolved problems by training it to match a small and narrow set of oracle solutions can inadvertently damage its ability to solve other problems. This is because forcing the model to imitate oracle traces from a different distribution μ , outside the support of its own rollouts, distorts the next-token distribution produced by the fine-tuned model on other prefixes. We illustrate this idea in Figure 5. This effect has been documented in prior work (Kang et al., 2024; Setlur et al., 2025a), where training on off-policy traces induced memorization and catastrophic forgetting of base model skills¹.

In contrast, **InT** only clones single-step off-policy interventions, with the rest of the target sequence coming from a model-generated rollouts. Cloning behavior already produced by the base model primarily sharpens the next-token distribution on observed prefixes, without broadly distorting other conditionals. Although cloning the intervention conditioned on the preceding prefix could, in principle, distort the next token distributions akin to cloning an entire oracle trace discussed above, our interventions are only a few tokens long, making any such adverse impact far more limited.

6 EXPERIMENTS

The goal of our experiments is to evaluate the efficacy of **InT** in patching model behavior on hard training problems. In particular, we are interested in answering the following questions: (1) does SFT on just a few tokens of step-level oracle interventions improve the ability of the fine-tuned model to sample correct traces on hard problems? and (2) how does **InT** compare with distillation of full expert reasoning traces sampled from the oracle? To this end, we run several experiments comparing **InT** against running standard RL training and distillation on oracle solutions, in an attempt to patch the capabilities of e3-1.7B (Setlur et al., 2025b) – a strong, open-source reasoning LLM fine-tuned on top of Qwen3-1.7B – on a set of difficult math reasoning problems.

6.1 EXPERIMENTAL SETUP AND EVALUATION METRICS

Constructing a dataset of hard training problems. We begin our experiments with a state-of-the-art $<2\text{B}$ parameter model, e3-1.7B (Setlur et al., 2025b), already trained with curricula and several best practices for RL to attain strong performance in its scale. Despite its strong performance, this model still fails on a large fraction of problems from its hard training set (a 2.5K subset of DeepScaleR problems from Luo et al. (2025)). To isolate problems with zero rewards, we run 32 rollouts on each and collect the subset of problems the model cannot solve at all. We utilize Gemini 2.5 Pro (as of 2025-08-01) as our oracle. Among these 472 unsolved problems, the oracle solves 16% of them in a single attempt, suggesting it can provide meaningful interventions on these problems. We retain this subset as our hard problem set $\mathcal{D}_{\text{hard}}$ to study the efficacy of patching with different methods. Our main findings are that (i) **RL with just a small dataset of 64 problems on top of InT outperforms**

¹Mid-training typically runs behavior cloning (BC) to instill basic reasoning skills (DeepSeek-AI et al., 2025; Wang et al., 2025), using large, diverse datasets on pre-trained base models. In contrast, we address the challenge of solving difficult problems from only a small number of oracle traces – a setting in which BC is ineffective.

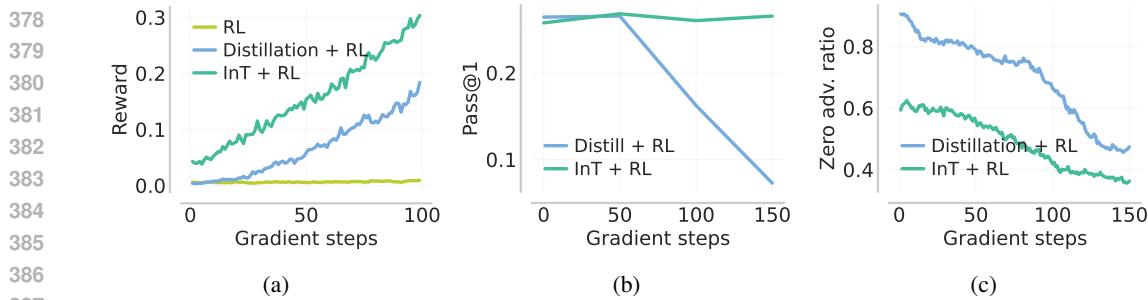


Figure 8: **Comparison of InT with distillation of oracle full length reasoning traces:** (a) Since these are hard problems, running RL initialized from the base model does not improve training reward, while running RL on top of the distilled model or the patched model produced by InT does improve training reward. We observe that running RL on top of the distilled model degrades model capability (decreasing pass@1 score on a held-out set in (b) as training progresses), even though distillation continues to make progress on the training set, as indicated by a decreasing ratio of the percentage of unsolved problems (“zero advantage ratio”) in (c).

RL on a much larger set of 1.2K problems on top of distillation or the base model. On the other hand, (ii) RL with the small dataset on top of the distillation and the base model are infeasible due to collapse of behaviors on OOD sets or zero learning signals. We also run some of our experiments on the Qwen3-4B-Instruct model, and we will present results with that model below.

Baselines approaches and comparisons. To evaluate the efficacy of **InT**, we compare against alternate approaches for patching model behavior on $\mathcal{D}_{\text{hard}}$. Our primary comparisons are: 1) “**Distillation + RL**,” which first distills entire oracle solutions into the base model before running RL, and 2) “**Standard RL**,” which directly continues RL on the hard problem set from the same base checkpoint. Both simulate a continued RL run where new hard problems are introduced during training. We also consider SFT-only baselines, where the model is patched via supervised learning on oracle solutions or intervention traces for the hard problems, without any further RL. To our knowledge, no existing method is designed to explicitly handle this setting of patching model behavior on previously unsolved hard problems in a way that leverages oracle interventions while preserving the benefits of RL. Therefore, we limit our exposition of training trends to $\mathcal{D}_{\text{hard}}$, but also compare with alternate approaches for using intervention data on holdout standardized test sets.

Evaluation metrics. Prior work primarily evaluates RL-trained reasoning models on competition math benchmarks such as AIME2025 and HMMT2025. However, progress on these alone does not capture whether models are actually learning from hard training problems, nor whether such training transfers to equally challenging evaluation problems. To address this, we evaluate our patched models on several standardized benchmarks from 2025, including AIME2025, HMMT2025, BRUMO2025, CMIMC2025, and others, as well as an in-distribution (i.i.d.) test set of hard problems $\mathcal{D}_{\text{hard}}^{\text{test}}$, similar to $\mathcal{D}_{\text{hard}}$. The i.i.d. test set consists of 64 problems held out from the training pool using the same methodology as used to select $\mathcal{D}_{\text{hard}}$. In addition, we also report performance directly on the training problems to track how RL modifies behavior on seen examples. Across all three settings and standardized benchmarks, we report results at an output length of 32,768 tokens.

6.2 INT UNIFORMLY PUSHES THE PASS@ k FRONTIER UPWARDS ON TEST PROBLEMS

We present our main results for InT on an holdout set of hard problems $\mathcal{D}_{\text{hard}}^{\text{test}}$ (Fig 7). Here, we plot the pass@ k performance across different RL training iterations from 0 to 150, for three models: (i) base e3-1.7B, (ii) e3-1.7B distilled on full oracle traces; and (iii) e3-1.7B patched on interventions from the oracle (InT). We find that running RL on the base or distilled model does not make any improvements in pass@ k throughout all training steps. On the other hand, running RL on $\mathcal{D}_{\text{hard}}$ after we patch e3-1.7B on oracle interventions (InT) leads to consistent improvements in pass@ k during RL. On training problems in $\mathcal{D}_{\text{hard}}$ running SFT on oracle interventions consistently improves performance across multiple problems (Sec 5), and running RL on $\mathcal{D}_{\text{hard}}$ with this initialization no longer leads to severe sharpening that we see when we run RL with the base or distilled models where the performance across problems $\mathcal{D}_{\text{hard}}$ is quite disparate for the RL initialization.

6.3 INT OUTPERFORMS DISTILLATION ON STANDARDIZED EVALUATIONS

Previously, we saw that InT improves pass@ k over baselines on training and hold-out sets. This mainly tells us that InT makes progress on the hard training problems that were previously unsolved.

432 But, we also care about how this gain in performance translates to performance on standardized
 433 benchmarks for math reasoning. Here, we compare the performance of our approach InT on top of
 434 the e3-1.7B reasoning model, and also the Qwen3-4B instruct model.

435 To stress test InT in the setting where we simply continue to run RL from the intervention checkpoint,
 436 we run RL training on this checkpoint with only 64 problems in $\mathcal{D}_{\text{hard}}$, on which we collected the
 437 interventions. We compare the performance of this RL trained model with an RL run on the distilled
 438 and base models. To boost the baselines, we run RL for both using an expanded set of about 1.3k
 439 problems sourced from DAPO (Yu et al., 2025), including the 64 we used for InT. The main reason
 440 we perform this injection is that in our preliminary experiments which trained the distilled model
 441 only on the small set of 64, we noticed that post RL the model capabilities on standardized evals
 442 fell drastically (Figure 8(b)), perhaps due to memorization and overfitting issues with the distilled
 443 model that we discussed in Section 5. When we run RL on the base model, we also expand the
 444 training set for RL, since we find that the reward curve does not rise otherwise (Figure 8(a))—thus
 445 we train the base model on a mixture of easy problems from DAPO and the 64 problems in InT
 446 dataset. Unlike the RL runs on distilled and base checkpoints, **InT improves the test performance**
 447 **averaged across multiple hard test datasets, despite being trained on just 64 problems** (Table 2).
 448 Compared to distillation, we see gains on both in-distribution ($\mathcal{D}_{\text{hard}}^{\text{test}}$) and standardized benchmarks
 449 for hard problems mainly because intervention does not alter the base model distribution as much as
 450 distillation (InT only SFTs on very few tokens in the intervention data, compared to distillation).

Model	RL Data Size	OlymMATH		OlymMATH	
		Easy	Hard	HMML	BRUMO
e3-1.7B + RL	1216	38.75	6.75	22.50	46.25
e3-1.7B + Distill + RL	1216	37.38	5.75	22.50	47.08
e3-1.7B + InT+ RL	64	41.62	7.50	24.58	53.75
Qwen3-4B-Inst + RL	1447	56.62	11.50	30.00	57.92
Qwen3-4B-Inst + Distill + RL	1447	56.12	10.62	29.17	57.08
Qwen3-4B-Inst + InT + RL	295	55.12	9.75	30.83	56.67
Model	AIME	Beyond AIME		CMIMC	Average
					$D_{\text{hard}}^{\text{test}}$ pass@8
e3-1.7B + RL	36.25	20.88		23.75	27.73
e3-1.7B + Distill + RL	36.67	21.75		21.56	27.38
e3-1.7B + InT+ RL	36.25	22.00		21.88	29.65
Qwen3-4B-Inst + RL	43.75	32.00		31.56	37.62
Qwen3-4B-Inst + Distill + RL	50.00	31.25		30.63	37.84
Qwen3-4B-Inst + InT + RL	43.33	30.38		29.06	36.45
					14.66

466 Table 2: Pass@1 performance (8 rollouts avg.) of models across standard mathematics benchmarks and pass@8
 467 performance on the i.i.d. test set, $D_{\text{hard}}^{\text{test}}$. Observe that InT followed by RL attains the highest pass@8 performance
 468 on this in-distribution test set for both patching the e3-1.7B base model as well as the Qwen3-4B-Instruct model.

499 7 DISCUSSION AND FUTURE WORK

500 In this work, we introduced **InT**, a simple yet effective approach for enabling continued RL training of
 501 reasoning LLMs by patching the base model with oracle-generated intervention data. Our motivation
 502 stems from the observation that a substantial fraction of failures on complex tasks arise from
 503 execution errors, cases where one or a few missteps derail the entire solution, leaving the model with
 504 no positive reward signal. **InT** addresses this challenge through targeted credit assignment: at the
 505 first mistake, an oracle provides a corrective intervention, and we fine-tune the model on the prefix
 506 and intervention trace. The resulting patched model can then resume RL training on problems that
 507 previously yielded no learning signal, extending the reach of RL beyond its traditional limits. **Moving**
 508 **forward, generalizing InT to open-ended reasoning** (e.g., IMO-ProofBench), **symbolic tasks**, and
 509 **subjective domains** is a natural next step, and we plan to explore these directions in future work.
 510 **Scaling InT to larger models and even harder domains such as FrontierMath and HLE** also present
 511 exciting directions for future work.

512 8 REPRODUCIBILITY STATEMENT

513 We have taken extensive measures to ensure that our results are reproducible. A detailed description of
 514 the proposed method, including the InT protocol, data collection steps, supervised fine-tuning setup,

486 and continuation of RL post-training, is provided in the main text (Secs. 3–6) and Algorithm 1. The
 487 construction of the hard problem set, evaluation metrics, and baseline comparisons are described in
 488 Sec. 6.1. Additional experimental details, ablations, and prompt templates for generating interventions
 489 are included in the appendices (Apps. A–E). We also report $\text{pass}@k$ metrics, bootstrapped confidence
 490 intervals, and benchmark evaluations across both in-distribution and out-of-distribution settings (Secs.
 491 6.2–6.3). These resources should enable independent researchers to replicate and extend both the
 492 empirical and methodological findings of this paper.

493

494

REFERENCES

495

496 Mohammad Hossein Amani, Aryo Lotfi, Nicolas Mario Baldwin, Samy Bengio, Mehrdad Farajtabar,
 497 Emmanuel Abbe, and Robert West. Rl for reasoning by adaptively revealing rationales, 2025. URL
 498 <https://arxiv.org/abs/2506.18110>.

499 Shengnan An, Xunliang Cai, Xuezhi Cao, Xiaoyu Li, Yehao Lin, Junlin Liu, Xinxuan Lv, Dan Ma,
 500 Xuanlin Wang, Ziwen Wang, and Shuang Zhou. Amo-bench: Large language models still struggle
 501 in high school math competitions, 2025. URL <https://arxiv.org/abs/2510.26768>.

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

504 Angelica Chen, Jérémie Scheurer, Jon Ander Campos, Tomasz Korbak, Jun Shern Chan, Samuel R.
 505 Bowman, Kyunghyun Cho, and Ethan Perez. Learning from natural language feedback. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL <https://openreview.net/forum?id=xo3hI5MwvU>.

506 DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu,
 507 Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu,
 508 Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao
 509 Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan,
 510 Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao,
 511 Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding,
 512 Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang
 513 Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong,
 514 Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao,
 515 Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang,
 516 Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang,
 517 Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L.
 518 Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang,
 519 Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng
 520 Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng
 521 Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan
 522 Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang,
 523 Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen,
 524 Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li,
 525 Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang,
 526 Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan,
 527 Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia
 528 He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong
 529 Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha,
 530 Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang,
 531 Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li,
 532 Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen
 533 Zhang. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025.
 534 URL <https://arxiv.org/abs/2501.12948>.

535 Bofei Gao, Feifan Song, Zhe Yang, Zefan Cai, Yibo Miao, Qingxiu Dong, Lei Li, Chenghao Ma,
 536 Liang Chen, Runxin Xu, Zhengyang Tang, Benyou Wang, Daoguang Zan, Shanghaoran Quan,

540 Ge Zhang, Lei Sha, Yichang Zhang, Xuancheng Ren, Tianyu Liu, and Baobao Chang. Omni-
 541 math: A universal olympiad level mathematic benchmark for large language models, 2024. URL
 542 <https://arxiv.org/abs/2410.07985>.

543

544 Zheyuan Hu, Robyn Wu, Naveen Enock, Jasmine Li, Riya Kadakia, Zackory Erickson, and Aviral
 545 Kumar. Rac: Robot learning for long-horizon tasks by scaling recovery and correction, 2025. URL
 546 <https://arxiv.org/abs/2509.07953>.

547

548 Katie Kang, Amrith Setlur, Dibya Ghosh, Jacob Steinhardt, Claire Tomlin, Sergey Levine, and Aviral
 549 Kumar. What do learning dynamics reveal about generalization in llm reasoning?, 2024. URL
 550 <https://arxiv.org/abs/2411.07681>.

551

552 Amirkhossein Kazemnejad, Milad Aghajohari, Eva Portelance, Alessandro Sordoni, Siva Reddy,
 553 Aaron Courville, and Nicolas Le Roux. Vineppo: Unlocking rl potential for llm reasoning through
 554 refined credit assignment. *arXiv preprint arXiv:2410.01679*, 2024.

555

556 Michael Kelly, Chelsea Sidrane, Katherine Driggs-Campbell, and Mykel J. Kochenderfer. Hg-dagger:
 557 Interactive imitation learning with human experts, 2019. URL <https://arxiv.org/abs/1810.02890>.

558

559 Muhammad Khalifa, Rishabh Agarwal, Lajanugen Logeswaran, Jaekyeom Kim, Hao Peng, Moontae
 560 Lee, Honglak Lee, and Lu Wang. Process reward models that think, 2025. URL <https://arxiv.org/abs/2504.16828>.

561

562 Seungone Kim, Ian Wu, Jinu Lee, Xiang Yue, Seongyun Lee, Mingyeong Moon, Kiril Gashtelovski,
 563 Carolin Lawrence, Julia Hockenmaier, Graham Neubig, and Sean Welleck. Scaling evaluation-time
 564 compute with reasoning models as process evaluators, 2025. URL <https://arxiv.org/abs/2503.19877>.

565

566 Aviral Kumar, Vincent Zhuang, Rishabh Agarwal, Yi Su, John D Co-Reyes, Avi Singh, Kate Baumli,
 567 Shariq Iqbal, Colton Bishop, Rebecca Roelofs, et al. Training language models to self-correct via
 568 reinforcement learning. *arXiv preprint arXiv:2409.12917*, 2024.

569

570 Jiazheng Li, Hongzhou Lin, Hong Lu, Kaiyue Wen, Zaiwen Yang, Jiaxuan Gao, Yi Wu, and Jingzhao
 571 Zhang. Questa: Expanding reasoning capacity in llms via question augmentation, 2025. URL
 572 <https://arxiv.org/abs/2507.13266>.

573

574 Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan
 575 Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *arXiv preprint*
 576 *arXiv:2305.20050*, 2023a.

577

578 Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike,
 579 John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step, 2023b.

580

581 Zijun Liu, Peiyi Wang, Runxin Xu, Shirong Ma, Chong Ruan, Peng Li, Yang Liu, and Yu Wu.
 582 Inference-time scaling for generalist reward modeling, 2025. URL <https://arxiv.org/abs/2504.02495>.

583

584 Kevin Lu and Thinking Machines Lab. On-policy distillation. *Thinking Machines Lab: Connectionism*, 2025. doi: 10.64434/tm1.20251026. <https://thinkingmachines.ai/blog/on-policy-distillation>.

585

586 Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Meiqi Guo, Harsh Lara, Yunxuan Li, Lei
 587 Shu, Yun Zhu, Lei Meng, Jiao Sun, and Abhinav Rastogi. Improve mathematical reasoning in
 588 language models by automated process supervision, 2024. URL <https://arxiv.org/abs/2406.06592>.

589

590 Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Y. Tang, Manan Roongta, Colin Cai,
 591 Jeffrey Luo, Li Erran Li, Raluca Ada Popa, and Ion Stoica. Deepscaler: Surpassing o1-preview
 592 with a 1.5b model by scaling rl. <https://pretty-radio-b75.notion.site/DeepScaler-Surpassing-O1-Preview-with-a-1-5B-Model-by-Scaling-RL-19681902c1468005bed8ca303013a4e2>, 2025. Notion Blog.

594 Thang Luong, Dawsen Hwang, Hoang H. Nguyen, Golnaz Ghiasi, Yuri Chervonyi, Insuk Seo, Junsu
 595 Kim, Garrett Bingham, Jonathan Lee, Swaroop Mishra, Alex Zhai, Clara Huiyi Hu, Henryk
 596 Michalewski, Jimin Kim, Jeonghyun Ahn, Junhwi Bae, Xingyou Song, Trieu H. Trinh, Quoc V.
 597 Le, and Junehyuk Jung. Towards robust mathematical reasoning. In *Proceedings of the 2025*
 598 *Conference on Empirical Methods in Natural Language Processing*, 2025. URL <https://aclanthology.org/2025.emnlp-main.1794/>.

600 Yuxiao Qu, Tianjun Zhang, Naman Garg, and Aviral Kumar. Recursive introspection: Teaching
 601 language model agents how to self-improve. *arXiv preprint arXiv:2407.18219*, 2024.

602 Yuxiao Qu, Anikait Singh, Yoonho Lee, Amirth Setlur, Ruslan Salakhutdinov, Chelsea Finn, and
 603 Aviral Kumar. Learning to discover abstractions for LLM reasoning. In *ICML 2025 Workshop on*
 604 *Programmatic Representations for Agent Learning*, 2025a. URL <https://openreview.net/forum?id=zwEU0OKT8G>.

605 Yuxiao Qu, Matthew YR Yang, Amirth Setlur, Lewis Tunstall, Edward Emanuel Beeching, Ruslan
 606 Salakhutdinov, and Aviral Kumar. Optimizing test-time compute via meta reinforcement fine-
 607 tuning. *arXiv preprint arXiv:2503.07572*, 2025b.

608 Amirth Setlur, Saurabh Garg, Xinyang Geng, Naman Garg, Virginia Smith, and Aviral Kumar. RL on
 609 incorrect synthetic data scales the efficiency of llm math reasoning by eight-fold. *arXiv preprint*
 610 *arXiv:2406.14532*, 2024a.

611 Amirth Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal,
 612 Alekh Agarwal, Jonathan Berant, and Aviral Kumar. Rewarding progress: Scaling automated
 613 process verifiers for llm reasoning. *arXiv preprint arXiv:2410.08146*, 2024b.

614 Amirth Setlur, Nived Rajaraman, Sergey Levine, and Aviral Kumar. Scaling test-time compute without
 615 verification or rl is suboptimal, 2025a. URL <https://arxiv.org/abs/2502.12118>.

616 Amirth Setlur, Matthew Y. R. Yang, Charlie Snell, Jeremy Greer, Ian Wu, Virginia Smith, Max
 617 Simchowitz, and Aviral Kumar. e3: Learning to explore enables extrapolation of test-time compute
 618 for llms, 2025b. URL <https://arxiv.org/abs/2506.09026>.

619 Idan Shenfeld, Jyothish Pari, and Pulkit Agrawal. RL's razor: Why online reinforcement learning
 620 forgets less, 2025. URL <https://arxiv.org/abs/2509.04259>.

621 Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng,
 622 Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint*
 623 *arXiv: 2409.19256*, 2024.

624 Akshit Sinha, Arvindh Arun, Shashwat Goel, Steffen Staab, and Jonas Geiping. The illusion of
 625 diminishing returns: Measuring long horizon execution in llms, 2025. URL <https://arxiv.org/abs/2509.09677>.

626 Peiyi Wang, Lei Li, Zhihong Shao, R. X. Xu, Damai Dai, Yifei Li, Deli Chen, Y. Wu, and Zhifang
 627 Sui. Math-shepherd: Verify and reinforce llms step-by-step without human annotations, 2024.

628 Zengzhi Wang, Fan Zhou, Xuefeng Li, and Pengfei Liu. Octothinker: Mid-training incentivizes
 629 reinforcement learning scaling, 2025. URL <https://arxiv.org/abs/2506.20512>.

630 Jianhao Yan, Yafu Li, Zican Hu, Zhi Wang, Ganqu Cui, Xiaoye Qu, Yu Cheng, and Yue Zhang.
 631 Learning to reason under off-policy guidance, 2025. URL <https://arxiv.org/abs/2504.14945>.

632 Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian
 633 Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at
 634 scale. *arXiv preprint arXiv:2503.14476*, 2025.

635 Lunjun Zhang, Arian Hosseini, Hritik Bansal, Mehran Kazemi, Aviral Kumar, and Rishabh Agarwal.
 636 Generative verifiers: Reward modeling as next-token prediction. *arXiv preprint arXiv:2408.15240*,
 637 2024.

648 Wenhao Zhang, Yuexiang Xie, Yuchang Sun, Yanxi Chen, Guoyin Wang, Yaliang Li, Bolin Ding,
649 and Jingren Zhou. On-policy rl meets off-policy experts: Harmonizing supervised fine-tuning and
650 reinforcement learning via dynamic weighting, 2025a. URL <https://arxiv.org/abs/2508.11408>.

651
652 Xiaoying Zhang, Hao Sun, Yipeng Zhang, Kaituo Feng, Chaochao Lu, Chao Yang, and Helen Meng.
653 Critique-grpo: Advancing llm reasoning with natural language and numerical feedback, 2025b.
654 URL <https://arxiv.org/abs/2506.03106>.

655
656 Xuechen Zhang, Zijian Huang, Yingcong Li, Chenshun Ni, Jiasi Chen, and Samet Oymak. Bread:
657 Branched rollouts from expert anchors bridge sft rl for reasoning, 2025c. URL <https://arxiv.org/abs/2506.17211>.

658
659 Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyuan Luo, Zhangchi Feng, and
660 Yongqiang Ma. Llamafactory: Unified efficient fine-tuning of 100+ language models. In *Pro-
661 ceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3:
662 System Demonstrations)*, Bangkok, Thailand, 2024. Association for Computational Linguistics.
663 URL <http://arxiv.org/abs/2403.13372>.

664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701

756

757 ****4.3 Replacement Steps List****758 Here you should summarize the list of potential recovery locations and steps. Please write
759 the steps from the student's perspective. The student should continue from your step without
760 feeling that someone else wrote it.761 Finally, include a final curated list of Replacement Steps List to be processed in a parser. This
762 list should strictly follow the format below with only a number at the step number, and the
763 replacement step afterwards. ****DO NOT INCLUDE ADDITIONAL JUSTIFICATIONS, OR**
ELSE THE PARSER CAN'T PARSE CORRECTLY.**764 If you believe that the student's solution is on the right track and there are no critical errors,
765 leave the list empty.766 ****Format:****767 <replacement> [{ "location": INSERT_STEP_NUMBER, "content": IN-
768 SERT_STEP_CONTENT }, ... { "location": INSERT_STEP_NUMBER, "content": IN-
769 SERT_STEP_CONTENT },] </replacement>770 **### Student Solution ###**

771 {Insert student solution}

772

773 **Prompt for error categorization**774 Here is a faulty student solution to a problem and a teacher's overview of the solution.
775 Please classify the primary mistake in the 'Student solution' as either a **High-Level Mistake**
776 (**Knowledge Gap**) or an **Execution-Level Mistake (Flawed Application)**, using the 'Teacher
777 guidance' for context. A **High-Level Mistake** means the model lacked fundamental knowl-
778 edge, used a completely wrong approach, or didn't know a key theorem. An **Execution-Level**
779 **Mistake** means the model had the right idea and a valid strategy but made a specific error
780 during its execution, such as a calculation slip or misapplying a known theorem.

781 Student: {insert solution}

782 Teacher: {insert guidance}

783 Give your response by saying either 'HIGH' or 'EXECUTION'.

784

785

786

787

788

789

790

791

792

793

794

795

796

797

798

799

800

801

802

803

804

805

806

807

808

809

810 B INTERVENTION EXAMPLES

811
812 In this section, we provide a few concrete examples of interventions.
813
814

815 **Problem Statement: Regular Polygon with Equilateral Triangle**

816 Point B is in the exterior of the regular n -sided polygon $A_1 A_2 \dots A_n$, and $A_1 A_2 B$ is an equilateral triangle. What is the largest
817 value of n for which A_1 , A_n , and B are consecutive vertices of a regular polygon?
818
819

820 **Steps 38-40: Shared Context**

821 **Step 38:** We know:
822 **Step 39:** $-A_1 = 1 \rightarrow \arg = 0 - A_n = \omega^{-1} = e^{-2\pi i/n} \rightarrow$
823 $\arg = -2\pi/n - B = 1 + (\omega - 1)e^{-i\pi/3}$
824 **Step 40:** We want A_1 , A_n , B to be **consecutive vertices** of a regular polygon.
825
826

827 **Steps 41-44: Student's Wrong Path**

828 **Step 41:** So the angular differences between them must
829 be equal.
830 **Step 42:** That is:
831

$$832 \arg(A_n) - \arg(A_1) = \arg(B) - \arg(A_n)$$

833 Let's denote $\theta = \arg(B) - \arg(A_n)$

834 **Step 43:** Then:

$$835 \arg(A_n) - \arg(A_1) = \theta$$

$$836 \arg(B) - \arg(A_n) = \theta$$

837 So:

$$838 \arg(B) = 2\arg(A_n) - \arg(A_1)$$

839 **Step 44:** Since $\arg(A_1) = 0$, $\arg(A_n) = -2\pi/n$, so:
840

$$\arg(B) = -4\pi/n$$

841 **Step 41: InT Guidance**

842 So the distances between consecutive points must be equal.
843 Let's consider the possible orderings. If the order is
844 A_n, A_1, B , then we must have $|A_n - A_1| = |A_1 - B|$. Let's
845 calculate these lengths and see if this condition holds or
846 gives us constraints on n . Then, we must check the angle
847 $\angle A_n A_1 B$.
848

849 **Steps 43-44: Post-Correction**

850 **Step 43:** Step 3: Use rotational symmetry
851 **Step 44:** Suppose A_1, A_n, B are **consecutive vertices** of a regular m -gon.
852

853 **Final Answer: Student Gets Wrong**

854 5

855 **Final Answer: Model Gets Correct**

856 42

857 **Problem Statement: A given sequence r_1**

858 A given sequence r_1, r_2, \dots, r_n of distinct real numbers can be put in ascending order by means of one or more "bubble passes".
859 A bubble pass through a given sequence consists of comparing the second term with the first term, and exchanging them if and
860 only if the second term is smaller, then comparing the third term with the second term and exchanging them if and only if the
861 third term is smaller, and so on in order, through comparing the last term, r_n , with its current predecessor and exchanging
862 them if and only if the last term is smaller. The example below shows how the sequence 1, 9, 8, 7 is transformed into the
863 sequence 1, 8, 7, 9 by one bubble pass. The numbers compared at each step are underlined. 1 9 8 7 1 9 8 7 1 8 9 7
864 1 8 7 9 Suppose that $n = 40$, and that the terms of the initial sequence r_1, r_2, \dots, r_{40} are distinct from one another and
865 are in random order. Let p/q , in lowest terms, be the probability that the number that begins as r_{20} will end up, after one
866 bubble pass, in the 30th place. Find $p + q$.
867

868 **Steps 241-243: Shared Context**

869 **Step 241:** So, to stay at or move to a position 20, it must
870 not be smaller than any of the elements to its left.

871 **Step 242:** In particular, to **end up at position 30**, it must:

872 **Step 243:** 1. **Not be smaller than any of the elements** in positions 1 through 19 — otherwise it will be pushed
873 left.
874

875 **Steps 244-247: Student's Wrong Path**

876 **Step 244:** What if we use only numbers that are powers
877 of 2 and 3?

878 **Step 245:** Try numbers:

879 1, 2, 3, 4, 6, 8, 9, 12, 16, 18, 24, 27, 32, 36, 48, 54, 64, 72, 81, 96, 108, 128,
880 144, 162, 192, 216, 256, 288, 324, 384, 432, 512
881 ...
882

883 **Final Answer: Student Gets Wrong**

885 11010049

886 **Step 244: InT Guidance**

887 2. **Be larger than the elements in positions 21 through
888 30, and smaller than the element in position 31** — so
889 that it moves right by exactly 10 positions.
890

891 **Steps 245-247: Post-Correction**

892 **Step 245:** Wait — if it is larger than r_{21} , it will move to
893 21.
894 **Step 246:** If it is larger than r_{22} , it will move to 22.
895 **Step 247:** ...
896

897 **Final Answer: Model Gets Correct**

898 931

864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917**Problem Statement: Suppose Harvard Yard is a 17×17 ...**

Suppose Harvard Yard is a 17×17 square. There are 14 dorms located on the perimeter of the Yard. If s is the minimum distance between two dorms, the maximum possible value of s can be expressed as $a - \sqrt{b}$ where a, b are positive integers. Compute $100a + b$.

Steps 210-212: Shared Context

Step 210: So we need $d \geq s$
Step 211: Distance between a point on one side and a point on the adjacent side.
Step 212: For example, point at $(x, 0)$ on bottom and point at $(17, x)$ on right.

Steps 213-216: Student's Wrong Path

Step 213: Distance: $\sqrt{(17-x)^2 + (x-0)^2} = \sqrt{(17-x)^2 + x^2}$
Step 214: Similarly, between point at $(x, 0)$ and point at $(17, x+d)$ on right: distance $\sqrt{(17-x)^2 + (x+d)^2}$
Step 215: We want to maximize the minimal distance between any two points.
Step 216: By symmetry, the minimal distance will be the minimum of:

Final Answer: Student Gets Wrong

1819

Step 213: InT Guidance

Now, let's consider the distance between dorms on adjacent sides, which cross an unoccupied corner like $(17, 0)$. The last dorm on the bottom side is at $(3s, 0)$. By symmetry, the first dorm on the right side must be placed relative to the corner at $(17, 17)$ in the same way the first dorm on the bottom is placed relative to $(0, 0)$. The dorms on the right side are at distances $s, 2s, 3s$ from $(17, 17)$, so their coordinates are $(17, 17-s), (17, 17-2s), (17, 17-3s)$. The adjacent dorms across the corner $(17, 0)$ are therefore at $(3s, 0)$ and $(17, 17-3s)$.

Steps 214-216: Post-Correction

Step 214: $\sqrt{(17-3s)^2 + (17-3s)^2} = \sqrt{2}(17-3s)$
Step 215: We want this distance to be at least s .
Step 216: Similarly, the distance between adjacent points on the same side is $d = s$, so we need $d \geq s$ — trivial.

Final Answer: Model Gets Correct

602

Problem Statement: Polynomial Roots with Bounded Absolute Value

If x, y are real, then the *absolute value* of the complex number $z = x + yi$ is

$$|z| = \sqrt{x^2 + y^2}.$$

Find the number of polynomials $f(t) = A_0 + A_1t + A_2t^2 + A_3t^3 + t^4$ such that A_0, \dots, A_3 are integers and all roots of f in the complex plane have absolute value ≤ 1 .

Steps 109-111: Shared Context

...

Step 109: So for $\text{all roots to have } |z| \leq 1$, and for $\text{no root to have } |z| > 1$, then any root z with $|z| < 1$ must have its reciprocal $1/\bar{z}$ also a root — but $|1/\bar{z}| > 1$, contradiction.

Step 110: Therefore, $\text{no root can have } |z| < 1$ unless its reciprocal is also a root — but then $|z| > 1$, contradiction.

Step 111: Therefore, $\text{all roots must have } |z| = 1$.

Steps 112-114: Student's Wrong Path

Step 112: Wait — is that true?
Step 113: Yes! This is a key point.
Step 114: Key Theorem (in complex analysis):
 ...

Step 112: InT Guidance

Wait — is that true? Let's test this conclusion with a simple example. What about the polynomial $f(t) = t^4$? The coefficients are all integers ($A_3 = A_2 = A_1 = A_0 = 0$), and all its roots are $z = 0$. The absolute value is $|0| \leq 1$, so this polynomial should be counted. However, its roots are not on the unit circle. This means my conclusion that all roots must have $|z| = 1$ is incorrect. My reasoning in steps 108-111 must be flawed.

Steps 114-115: Post-Correction

Step 114: Roots can be inside the unit disk
Step 115: The key point is:
 ...

Final Answer: Student Gets Wrong

21

Final Answer: Student Gets Correct

43

918 **C TRAINING HYPERPARAMETERS**
919

Hyperparameter	e3-1.7B	Qwen3-4B-Instruct-2507
train_batch_size	32	32
ppo_mini_batch_size	16	16
learning_rate	1.0e-6	1.0e-6
kl_loss_coef	0.001	0.001
entropy_coeff	0.001	0
temperature	0.6	1.0
top_p	0.95	1.0
rollout.n	16	8
ppo_lowerclip_threshold	0.2	0.2
ppo_higherclip_threshold	0.35	0.35

931
932 Table 3: Verl [Sheng et al. \(2024\)](#) hyperparameters used for RL runs.
933

Hyperparameter	Distillation	InT
dataset_size	73	482
effective_batch_size	32	64
num_train_epochs	100	16
learning_rate	1.0e-7	1.0e-6
lr_scheduler_type	cosine_with_min_lr	cosine_with_min_lr
min_lr_rate	0.1	0.1
warmup_ratio	0.1	0.1

934
935 Table 4: LLaMa Factory [Sheng et al. \(2024\)](#) hyperparameters used for e3 SFT runs.
936

Hyperparameter	Distillation	InT
dataset_size	294	778
effective_batch_size	32	32
num_train_epochs	22	8
learning_rate	1.0e-7	1.0e-6
lr_scheduler_type	cosine_with_min_lr	cosine_with_min_lr
min_lr_rate	0.1	0.1
warmup_ratio	0.1	0.1

937
938 Table 5: LLaMa Factory [Zheng et al. \(2024\)](#) hyperparameters used for Qwen3-Instruct SFT runs.
939940 **D DATA COMPOSITION**
941942 For Qwen3-4B-Instruct, we take DAPO (14.1K English problems), DeepScaleR: (40.3k problems),
943 MathOdyssey (389 problems), Olympiad Bench (674 English, text only, Competition, Final Answer
944 problems), Putnam-AXIOM (492 problems), and filter down the hard problems for each model.
945946 **E EVAL CONFIGURATION**
947948 For e3-1.7B, we use a decoding setup with temperature 0.6, top-p 0.95, and top-k 20.
949950 For Qwen3-4B-Instruct, we follow the official recommended configuration, using temperature 0.7,
951 top-p 0.8, and top-k 20.
952953 **F USE OF LARGE LANGUAGE MODELS**
954955 We used large language models (LLMs) as an assistive tool primarily for rephrasing arguments more
956 crisply and for generating LaTeX templates (e.g., tables, algorithm boxes, or figure formatting). All
957

972 research ideas, developments, experiments, and empirical results were conceived, executed, and
 973 validated by the authors. The LLM did not contribute to the scientific content, claims, or findings of
 974 this work.

976 G IMPACT STATEMENT

977 This paper presents work whose goal is to advance the field of ML. There are many potential societal
 978 consequences of our work, none which we feel must be specifically highlighted here.

981 H SELF-GENERATED INTERVENTIONS WITHOUT AN EXTERNAL ORACLE

983 In this section, we describe **major changes** added to the **Int** pipeline by **removing the dependence**
 984 **on Gemini 2.5 Pro to instead use the base model to generate interventions**. This relaxes our
 985 assumptions of a stronger verifier model and shows that just by querying the base model itself to
 986 output interventions, **Int** allows the model to improve upon its own answers, perhaps by leveraging
 987 the difficulty gap between verification and generation in LLMs [Setlur et al. \(2025b\)](#).

988 **Scaled up the train set.** We scaled up our train set by filtering problems from the following sources:

- 990 1. Polaris (53k), filtering for problems that get zero accuracy under 64 rollouts.
- 991 2. AceReason-Math (50k), filtering for problems that get zero accuracy under 64 rollouts.
- 992 3. Omni-MATH (4.4k), filtering for problems that get zero accuracy under 128 rollouts.
- 993 4. IMO-AnswerBench (360), without the problems picked for the new test set.

995 After applying the difficulty filters, we end up
 996 with 4.5k problems. We then generate interventions
 997 on these problems and filter for interventions
 998 that lead to a non-zero reward at 32 rollouts
 999 and up with a set of 1076 problems with cor-
 1000 responding interventions. We take a subset of 334
 1001 problems and compare taking naive rollouts and
 1002 rolling out from the interventions. As shown
 1003 in Table 6, even without tuning the base model,
 1004 interventions-conditioned rollouts perform far
 1005 better than naive rollouts, achieving nonzero re-
 1006 wards on twice the problems and beating the accuracy by more than an order of magnitude.

1007 **Updated test sets.** To test our method on more difficult and standardized benchmarks, we leverage
 1008 IMO-Bench [Luong et al. \(2025\)](#), AMO-Bench [An et al. \(2025\)](#), Apex Shortlist [Balunović et al. \(2025\)](#),
 1009 some of which are released after the submission deadline. Additionally, we scraped the HMMT
 1010 2025 Novevember competition from HMMT’s official website, to test the models on brand new
 1011 problems formulated after the release of Qwen3-4B-Instruct to prevent train set contamination. For
 1012 IMO-Bench, we handpick 40 problems, 10 from each problem category (Algebra, Combinatorics,
 1013 Geometry, Number Theory), whose answers are easily verifiable (e.g., simple integers and fractions)
 1014 so that they are gauging the model’s Math ability rather than its ability to formulate expressions
 1015 that pass the particular parser. For the AMO-Bench problems, we apply the same filter and remove
 1016 problems whose answers that cannot be verified easily (e.g., paragraphs of proofs).

1017 **SFT in interventions and online RL.** We con-
 1018 tinute to run **Int** by performing SFT on these
 1019 interventions and subsequently online RL *for*
 1020 *much longer than before, up to 400 steps*. We
 1021 compare this against running RL on the base
 1022 model directly; on a SFT model trained on the
 1023 gold solutions; on a SFT model trained on self-
 1024 refinement (see details in Appendix J), and show
 1025 the training curves in Figure 9. We show the
 pass@k curves on the new test set throughout
 training in Figure 10, and our final results in Table 7.

Configuration	Nonzero Acc.	Accuracy
Naive	40/334	0.0984%
With intervention	80/334	1.56%

Table 6: **Accuracy of rollouts with or without conditioning on interventions.** As shown, rollouts conditioned on interventions double the number of problems with at least one correct rollout, and improve the rollout accuracy by more than an order of magnitude.

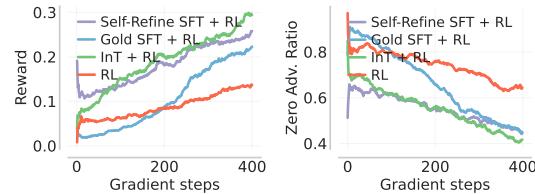


Figure 9: **Training reward and zero advantage ratio curves.** Zero advantage ratio curves represent the percentage of problems that the never succeeds on. ini

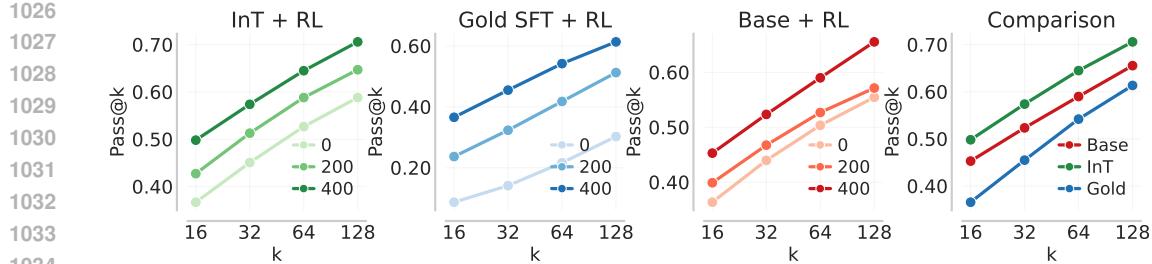


Figure 10: **Pass@k across RL training iterations:** We plot pass@ k performance from 0 to 400 RL iterations for three initializations: (i) base model patched with **InT**; (ii) base model SFT on gold traces; (iii) base model directly. **InT** achieves the highest pass@ k performance across all k .

Model	IMO-Bench	HMMT 2025 Nov	AMO-Bench pass@8	Apex Shortlist pass@8	Average
Base	11.68	41.61	26.24	20.79	25.08
+ RL	23.46	46.46	35.21	22.72	31.96
+ Gold SFT + RL	11.56	27.45	25.19	20.51	21.18
+ Self-refine SFT + RL	15.53	38.65	36.72	23.93	28.71
+ InT + RL (Ours)	25.62	49.77	36.16	28.22	34.94

Table 7: Pass@1 and pass@8 performance of different training methods estimated using 128 rollouts across difficult Math benchmarks. **InT** followed by RL attains the highest performance on most benchmarks. The base model used is on Qwen3-4B-Instruct.

I TRAINING ON OFF-POLICY TRACES IS INSUFFICIENT

In this section, we provide more evidence that demonstrate that training naively on off-policy traces may reduce performance on both training and test data after performing SFT, a phenomenon observed by concurrent works that also advocate for on-policy training [Lu & Lab \(2025\)](#); [Shenfeld et al. \(2025\)](#). We compare **InT** against SFT on the following baselines:

1. **Gold solutions**, which are mostly human-written solutions drawn from datasets such as DeepscaleR [Luo et al. \(2025\)](#) and Omni-MATH [Gao et al. \(2024\)](#), with a minority of the solutions drawn from Gemini 2.5 Pro.
2. **Self-refinement traces**, which are outputted by the base model Qwen3-4B-Instruct.
3. **Deepseek R1 traces**. We attempt to make these traces more on-policy by either only including the content between the thinking tags or the content after the closing think tag. This is done in attempt to match Qwen3-4B-Instruct’s traces, which only contains a single coherent block of reasoning in its output as opposed to the two-part format supported by Deepseek R1 [DeepSeek-AI et al. \(2025\)](#).

We run these baselines to cover three distinct “levels” of off-policiness: The first baseline mostly follows a human-written distribution; the second follows a base model self-generated distribution conditioned on a different self-refinement prompt; and the third follows an entirely LLM-generated distribution by a stronger external model.

In Figure 11 (a), we sample 64 random traces for each method (for the first baseline, we ensure that all 64 are human-written) and compute their negative log-likelihoods. We find that **InT** obtains the lowest negative log-likelihood among all methods, indicating the highest level of on-policiness. After performing SFT on these traces, we observe that **InT** obtains the highest pass@ k accuracies on both the train set and a held-out test set.

J SELF-REFINEMENT BASELINE

Self-refinement. We add details on how we establish the self-refinement baseline, which was of interest to several reviewers. The idea is similar to generating interventions, but rather than outputting single-step, oracle interventions, we ask the base model to re-write the entire solution.

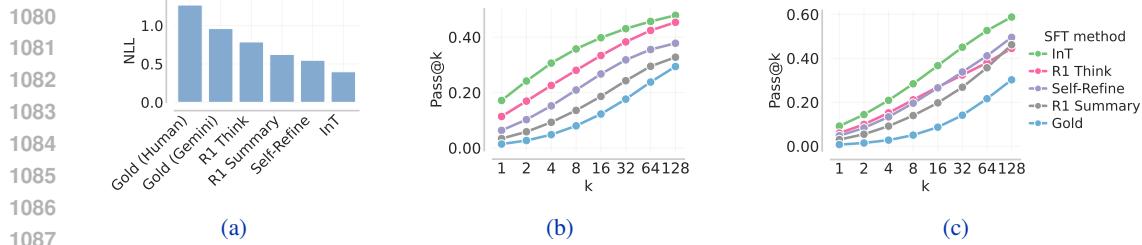


Figure 11: **InT facilitates on-policy learning.** (a) Negative log-likelihood (NLL) over 64 sampled traces. **Gold** denotes human-written solutions (with a small fraction of Gemini traces) taken directly from open-source datasets. **R1 Think** and **Summary** correspond to content inside and after DeepSeek R1 `<think>` tags. **InT** produces the most on-policy traces with the lowest NLL. (b) Train pass@k on 64 sampled training problems. **InT** achieves the highest pass@k across all k. (c) Test pass@k on IMO-Bench, AMO-Bench, and Apex Shortlist. **InT** again attains the best performance across all k.

We find that our method **InT** consistently outperforms the self-refinement baseline, as shown in Figure 10 and Table 7. The prompt for generating self-refinement traces is shown below. The gold solution is a high-level summary of the solution written by humans, and occasionally, a stronger model. Similar to **InT**, we ask the base model to generate its own self-refinements.

Prompt for self-refinement

You are an expert mathematician teaching a Math Olympiad class. You will be given a problem and a high-level gold solution to the problem. Your task is to solve the problem step-by-step guided by the high-level gold solution.

Problem #
 {Insert Problem}
 # High-Level Gold Solution #
 {Insert Gold Solution}
 {Insert Model Response}

Great job! Now, a student in your class has solved the problem incorrectly. You must leverage your understanding of the gold solution to rewrite a refined version of his attempt at the problem. **Your rewritten solution should be a complete solution to the problem. **

Incorrect Student Attempt #
 {Insert Student Solution}

K CONNECTION BETWEEN **INT** AND CREDIT ASSIGNMENT

One of the key requirements for successful RL training is good credit assignment: given a reward, we need some indication of how various actions (in LLM reasoning, this corresponds to the steps or tokens generated) contributed to its attainment. We believe that many difficult problems remain unsolvable by naive RL training because credit assignment is poor: despite receiving zero rewards on the problem, the model cannot identify the exact reasoning steps that contributed to the incorrect final answer. Instead, outcome-reward RL simply trains the model to downweight the entire response, potentially jeopardizing the model from attaining the right final answer ever again, because it is disincentivized from even producing the correct sets of steps that appeared in this trace which were downweighted due to some imperfect steps. This view is discussed in prior work Setlur et al. (2024a).

InT aims to ameliorate this concern by using a single-step oracle (a model with access to the gold solution) to detect and then correct these incorrect steps via injection of interventions (“patching”). By detecting the exact steps at which mistakes occur during training while reinforcing correct steps prior to the mistake, **InT** attributes zero rewards directly to the detected mistakes by upweighting counterfactual interventions. This allows the training procedure to localize attainment of reward to a particular step, and enables RL to hone in onto this step during training.

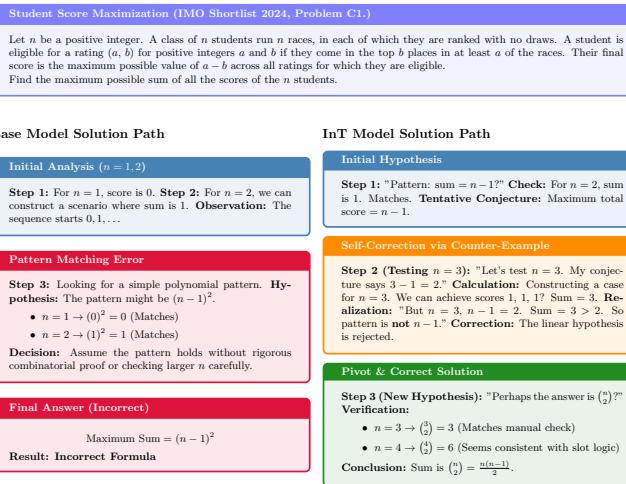
Although our approach is far from being the only way to do credit assignment, it offers a cheap and scalable solution by simply instructing the base model to propose an intervention given a gold human-written solution, which is widely available in math datasets. In the table above, we show that **InT** yields superior results when compared to standard RL and other baselines.

1134 L ARE INTERVENTIONS MEMORIZED?

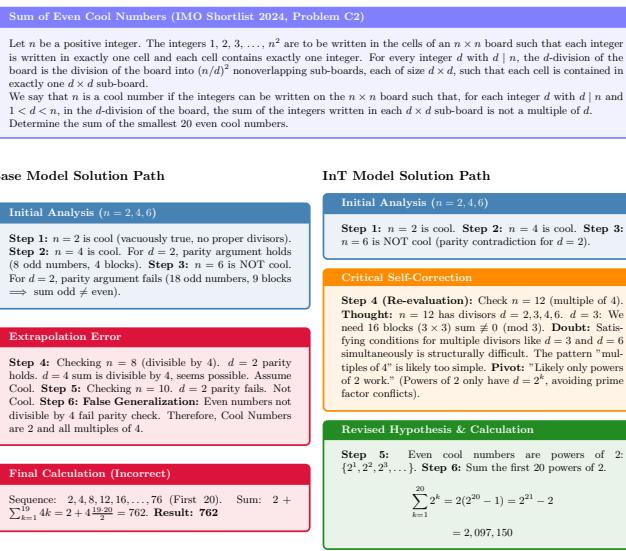
1135
 1136 We would like to understand whether **Int** leads to memorized interventions or if it actually learns to
 1137 generalize to *unseen* problems. As such, we select two example problems from the IMO Shortlist
 1138 2024 outside of our training set, and compare the traces of **Int** and base model (Qwen3-4B-Instruct).
 1139

1140 **IMO Shortlist 2024, Problem C1.** As shown in Figure 12, both models started with an incorrect
 1141 assumption of the formula, but when the **Int** trained model got 3 when $n = 3$, it questioned how this
 1142 was possible and thus correctly updated the hypothesis to $\binom{n}{2}$.
 1143

1144 **IMO Shortlist 2024, Problem C2.** Although both models are able to successfully conclude that even
 1145 cool numbers must be multiples of four in Figure 13, only the **Int** model is able to try considering
 1146 that $n = 12$ may not be cool and therefore the pattern may be more selective than simply being a
 1147 multiple of four. This leads the second model to the right hypothesis.
 1148



1166 Figure 12: Diverging Solution Paths between **Int** and base models on IMO Shortlist 2024, C1.
 1167



1187 Figure 13: Diverging Solution Paths between **Int** and base models on IMO Shortlist 2024, C2.

1188 M COMPARISON TO HINT-GUIDED RL METHODS

1189
 1190 In this section, we expand our positioning relative to contemporaneous approaches that combine RL
 1191 with oracle guidance or expert-driven hints. The central distinction is that none of these methods
 1192 performs *interventions* on the model’s own on-policy rollouts, whereas **Int** is explicitly built around
 1193 such targeted interventions. Another important distinction is that **Int** is evaluated on substantially
 1194 harder problem settings than those considered in prior work.

1195 **Interventions vs. Hints.** Recent methods such as QuestA (Li et al., 2025), AdaBack (Amani et al.,
 1196 2025), and BREAD (Zhang et al., 2025c) incorporate expert-generated guidance but only in the
 1197 form of *gold partial-solution prefixes*, i.e., static rationale snippets of gold solutions that condition
 1198 the model before rollout. None of these approaches modifies the model’s internal reasoning at the
 1199 point where its on-policy trajectory actually errors; their supervision remains entirely external to the
 1200 model’s own step-by-step reasoning. In contrast, **Int** is explicitly **interventional**: when a rollout
 1201 receives zero reward, **Int** identifies the first critical mistake within the model’s own trajectory and
 1202 injects a localized, natural-language correction at exactly that step. The model is then fine-tuned on
 1203 the corrected prefix before RL resumes, altering the causal structure of its on-policy reasoning and
 1204 providing more direct step-level credit assignment than approaches based solely on expert scaffolding.

1205 **Benchmark Difficulty.** Prior work evaluates primarily on benchmarks such as MATH, GSM8K,
 1206 NuminaMath-CoT, OpenR1-Math-220K, and mid-level competition suites including AIME, HMMT,
 1207 OlympiadBench, and BRUMO25, which are increasingly saturated for modern reasoning models.
 1208 In contrast, **Int** is additionally evaluated on substantially harder suites designed to probe frontier
 1209 reasoning limits: **IMO-Bench** (full IMO problems curated by medalists), **AMO-Bench** (IMO-level or
 1210 harder problems), and the **Apex Shortlist**, a collection of edge-of-capability problems where frontier
 1211 models achieve only around 50% accuracy. These benchmarks extend far beyond the difficulty of
 1212 those used in QuestA, AdaBack, or BREAD.

1213
 1214
 1215
 1216
 1217
 1218
 1219
 1220
 1221
 1222
 1223
 1224
 1225
 1226
 1227
 1228
 1229
 1230
 1231
 1232
 1233
 1234
 1235
 1236
 1237
 1238
 1239
 1240
 1241