Hide to Guide: Learning via Semantic Masking

Ruitao Liu[1]* **Qinghao Hu**[2]* **Alex Hu**[2] **Yecheng Wu**[3] **Shang Yang**[2]
Luke J. Huang[2] **Zhuoyang Zhang**[2] **Han Cai**[3] **Song Han**[2,3]

[1]Institute for Interdisciplinary Information Sciences, Tsinghua University
[2]Department of Electrical Engineering and Computer Science,
Massachusetts Institute of Technology [3]NVIDIA

Abstract

Reinforcement learning with verifiable rewards (RLVR) has become a powerful paradigm for improving language models on reasoning-intensive tasks, but its effectiveness is often limited by exploration. For example, models often fail on hard problems, leaving little useful reward signal. External expert traces offer a natural source of guidance, yet they may also expose reward-relevant content along the critical path to the verifier target, such as final answers, intermediate values, executable implementations, or answer-related entities. This content can create an unintended reward hacking channel, allowing the policy to obtain reward by copying the trace rather than learning the underlying reasoning or agentic behavior. Existing guided-RL methods reduce this risk by using partial trajectories, but they mainly control how much expert information is shown heuristically rather than which parts should be hidden. To this end, we propose Semantic Masked Expert Policy Optimization (SMEPO), a fine-grained semantic masking strategy for expert-guided RLVR. Instead of truncating traces coarsely or revealing them unchanged, SMEPO masks reward-relevant semantic spans along the critical path while preserving the expert's decomposition, plan, and procedural structure. This turns hard problems from reasoning from scratch into a fill-in-the-blank process: the policy can follow the expert's problem-solving route, but must still reconstruct the missing values, code, or entities by itself. SMEPO is simple to apply and requires no changes to the reward function or RL objective. Across diverse domains, including math, code, and agentic search, SMEPO improves accuracy by up to 3.2 points over GRPO and reduces training time by up to 4.2×. The code is available at `https://github.com/mit-han-lab/SMEPO`.

1 Introduction

Reinforcement learning with verifiable rewards (RLVR) has emerged as a prominent paradigm for enhancing large language models in complex domains such as mathematics, coding, and agentic search [1–7]. Unlike methods that rely on imitating static datasets, RLVR directly optimizes model behavior using automated verification. However, relying purely on self-exploration presents a critical bottleneck. On hard tasks, early rollouts frequently fail, resulting in sparse rewards that provide insufficient learning signals for the policy [8]. Consequently, practical RLVR systems typically require auxiliary guidance to steer the exploration process toward viable solution trajectories.

To provide this necessary guidance, a natural approach is to condition the RL process on expert traces, such as chain-of-thought rationales, solution trajectories, or hints. This concept has inspired a growing family of guided RL methods designed to enhance exploration [9–13]. While these approaches share the core intuition that expert signals can help policies navigate spaces they cannot explore autonomously, they notably avoid exposing the model to complete, unmodified expert traces. Instead, they rely on fragmented or altered signals, such as adaptive hints, partial prefixes, and

*Equal contribution.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

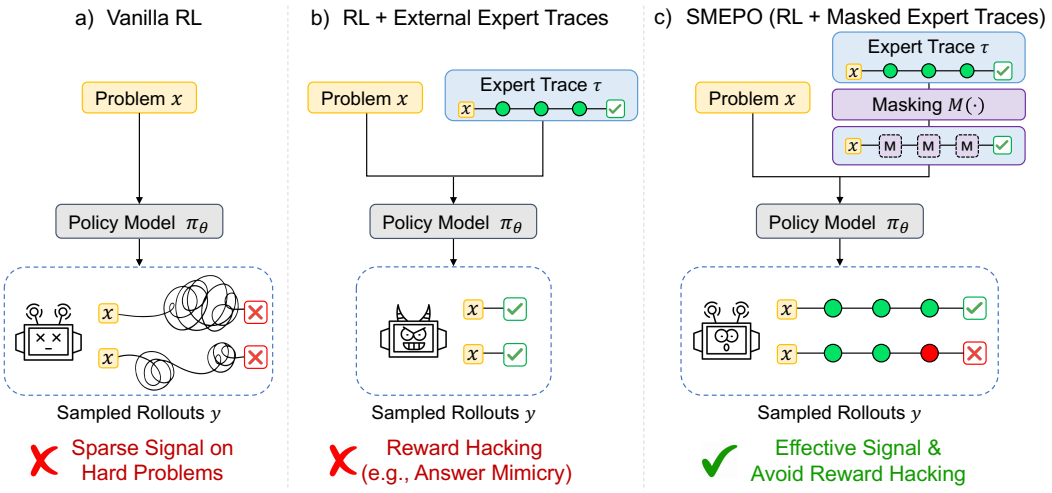


Figure 1: **Overview of SMEPO.** On hard reasoning problems, a) vanilla RL often receives sparse rewards because the initial policy fails to discover useful solution trajectories. b) Full expert traces can provide useful trajectories, but may also reveal verifier-aligned content along the solution path, such as intermediate values, final answers, code blocks, or answer-related entities. RL conditioned on the full trace may therefore exploit these exposed cues rather than learn the underlying reasoning process. c) SMEPO masks reward-relevant spans along the critical path while leaving the surrounding trace intact. The resulting fill-in-the-blank trace helps exploration while hiding verifier-aligned content.

rewritten trajectories. We argue that this deliberate design choice exposes a fundamental failure mode in standard expert-trace conditioning.

We attribute this failure to the dual nature of expert traces. They contain useful procedural cues, such as equation templates, solution plans, and query decomposition, but also expose reward-relevant content along the critical path to the verifier target. In math, this includes final answers and intermediate numerical values; in code, complete executable programs; and in agentic search, answer-relevant entities and query pivots. Since RLVR typically evaluates only the final outcome through domain-specific verifiers [1, 2, 4, 14], the policy may obtain reward by relying on these exposed cues rather than learning the reasoning, coding, or search process that produced them. In this way, full expert traces can turn guidance into an unintended reward hacking channel.

This perspective suggests a different design principle for guided RL: preserve the procedural scaffolding of expert traces while hiding verifier-aligned content along the critical path. Partial trajectories and adaptive hints can help because they reduce access to reward-relevant content, but when high-quality expert data is scarce, discarding large parts of the trace can also waste valuable guidance. In particular, it can remove or truncate the process-level information that makes expert traces useful, such as the step-by-step plan, intermediate subgoals, and problem-solving patterns.

We propose Semantic Masked Expert Policy Optimization (SMEPO), a semantically masked expert-guidance method that converts expert traces into fill-in-the-blank-style guidance. Instead of appending the full trace unchanged, SMEPO masks semantic spans that directly reveal the verifier-checked outcome or lie on the critical path to obtaining it, while preserving procedural guidance. We instantiate this principle in three representative RLVR domains, while the masking targets are not limited to these domains. For math, we mask numeric content in the expert trace; for code, we mask expert solution code; and for agentic search, we mask answer-relevant entities in expert search traces. The resulting trace preserves the expert's solution process, including decomposition, intermediate subgoals, and reasoning flow, while removing verifier-aligned content along the critical path. SMEPO introduces a guidance-level intervention: it reshapes the expert trace seen by the policy while keeping the verifier, reward function, and RL objective unchanged. This makes SMEPO easy to integrate into GRPO-style RLVR pipelines and combine with other guidance strategies.

Experiments show that SMEPO improves over both vanilla GRPO and direct expert conditioning. In our main 1k-example-per-domain setting, SMEPO improves Qwen3-8B-Base from 47.6 to 50.3 on

math, 56.5 to 59.0 on code, and 45.6 to 48.8 on agentic search. The gains transfer to additional math models, improving Qwen2.5-7B from 42.4 to 44.5 and DeepSeek-R1-Distill-Qwen-7B from 61.8 to 62.6, and SMEPO reaches hard-math targets earlier than vanilla GRPO. In a GHPO-based setting using the original GHPO data [9], adding SMEPO further improves the math average from 44.0 to 45.6. Diagnostic ablations support our explanation: random masking fails, fixed-prefix exposure is unstable, and progressively more targeted removal of reward-relevant content works best, indicating that the harmful signal is structured and reward-relevant rather than merely longer or noisier guidance.

We summarize our contributions as follows:

- We identify a critical failure mode in expert-guided RLVR. We characterize how policies bypass learning the underlying reasoning by exploiting verifier-aligned content (e.g., numerical values, entities) exposed in full expert traces across math, code, and agentic search domains.
- We propose SMEPO, a lightweight, guidance-level intervention. By masking reward-relevant spans while preserving the procedural scaffolding, SMEPO converts expert traces into effective fill-in-the-blank guidance without leaking solutions.
- We demonstrate that SMEPO significantly improves over GRPO and direct expert conditioning across multiple LLMs and diverse domains, while accelerating convergence on hard tasks.

2 Method

SMEPO is motivated by a tension in expert-guided RLVR: expert traces aid exploration but may also reveal content on the critical path to verifier success. Appendix D illustrates the two motivating failure modes: vanilla GRPO struggles to discover useful trajectories, whereas full-expert-trace conditioning exposes content that the policy can exploit during RL. We address this by separating reusable procedural guidance from reward-relevant content through lightweight semantic masking.

We consider reinforcement learning with verifiable rewards (RLVR). Given a problem prompt x, the policy π_θ generates a response y, and a domain-specific verifier assigns a reward $R(x, y)$. Unlike supervised imitation, the policy is optimized only through verifier feedback.

In expert-trace-guided RL, each training problem is additionally paired with an externally generated trace τ from a stronger model, such as a reasoning trace for math, a reference solution for code, or a search trajectory for agentic QA. A direct use of this trace is full-expert-trace conditioning:

$$y \sim \pi_\theta(\cdot \mid x, \tau), \tag{1}$$

while optimizing the same verifier reward $R(x, y)$. The trace is visible to the policy during rollout generation, but is not part of the generated answer; the verifier still scores only y. Thus, any benefit or harm from τ comes through how it changes rollout behavior.

This raises the central design question of SMEPO: how can we preserve the exploration benefits of expert traces while hiding reward-relevant content along the critical path? We formalize this tension by viewing an expert trace as containing two types of information:

$$\tau = \left(\tau^{\mathrm{g}}, \tau^{\mathrm{r}}\right). \tag{2}$$

Here τ^{g} denotes reusable guidance, such as solution organization, decomposition patterns, equation templates, control-flow logic, or step-by-step progression. In contrast, τ^{r} denotes content whose surface form is closely tied to verifier success, such as final answers, key intermediate values, executable implementations, or critical entities along a search path. When exposed unchanged, this reward-relevant content can become an unintended reward channel: the policy may rely on the trace to reach high verifier reward without learning the underlying reasoning, coding, or search process.

Motivated by this decomposition, SMEPO constructs a masked trace

$$\widetilde{\tau} = M(\tau), \tag{3}$$

where M preserves the order, formatting, and procedural organization of the trace while replacing reward-relevant spans with mask placeholders. The policy is then trained as

$$y \sim \pi_\theta(\cdot \mid x, \widetilde{\tau}), \tag{4}$$

using the same verifier reward $R(x, y)$ and the same GRPO pipeline as the baseline. Abstractly, this corresponds to optimizing

$$\max_\theta \ \mathbb{E}_{(x,\tau)\sim\mathcal{D},\ y\sim\pi_\theta(\cdot|x,\widetilde{\tau})}\left[R(x, y)\right]. \tag{5}$$

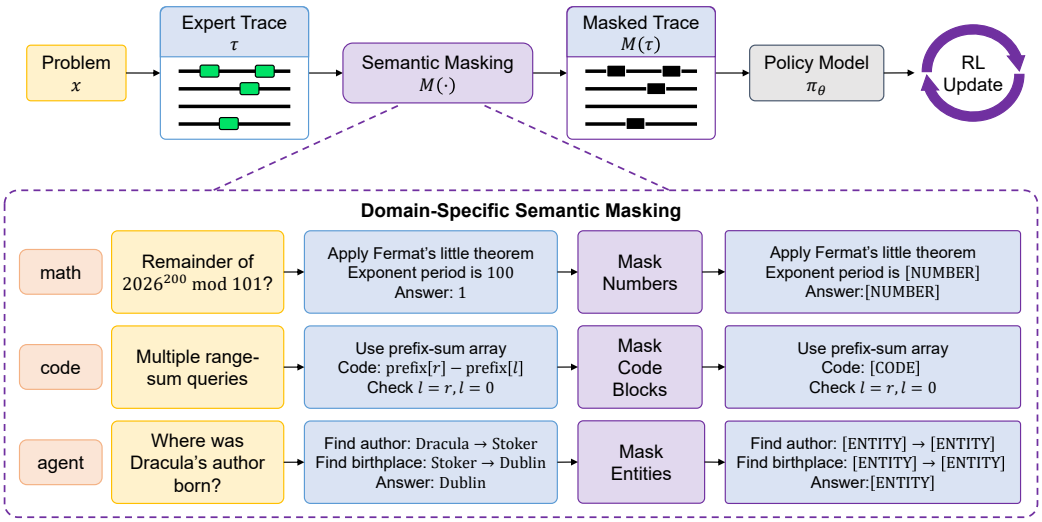


Figure 2: **Training pipeline for SMEPO.** We detect reward-relevant spans in the expert trace with domain-specific masking rules, replace those spans with mask placeholders, and then run the same GRPO loop conditioned on the original problem and the masked trace.

Figure 2 summarizes the workflow. SMEPO applies masking before policy optimization, so the learner receives a modified guidance trace throughout rollout generation. Although the concrete masking rules differ by domain, they follow the same principle: preserve expert guidance while suppressing content that either directly reveals the verifier target or exposes key intermediate information on the critical path to obtaining it.

Semantic masking. SMEPO masks reward-relevant content along the critical path to the verifier target while preserving the surrounding guidance. In RLVR, verifiers are domain-specific: math verifiers compare extracted numeric answers, code verifiers execute generated programs against tests, and agentic-search verifiers compare answer entities. SMEPO mirrors this verifier semantics at the guidance level: it masks semantic types that are directly checked by the verifier or reveal the path to what the verifier checks. Such spans can often be identified with lightweight domain-specific rules or semantic tools, such as number matching, code-block detection, named-entity recognition with spaCy [15], or lightweight general-purpose NER models such as GLiNER [16]. We instantiate this principle in three representative domains below, with additional implementation details in Appendix E.1.

Domain 1: math reasoning. For math, reward-relevant content often includes not only the final answer, but also repeated answer values and intermediate numeric values along the solution path. These numbers can reveal the target through copying, back-solving, or value-specific arithmetic. We therefore mask numeric spans in the expert trace, such as integers, decimals, and fractions. This preserves guidance such as decomposition steps, equation templates, and explanatory text, while requiring the policy to reconstruct the value-relevant parts of the solution itself.

Domain 2: code generation. For code, the most directly reward-relevant content is the expert program. A complete reference implementation can be copied, lightly adapted, or used as an executable template. We therefore mask expert code blocks while preserving surrounding non-code text. This removes exposed executable content while retaining useful guidance such as task interpretation, algorithmic plans, data-structure choices, corner-case discussions, and high-level debugging hints.

Domain 3: agentic search. For agentic search, expert trajectories can expose entities that are tightly coupled to verifier success. Expert trajectories may reveal intermediate entities, query pivots, supporting-page titles, and the final answer entity. Exposing these entities can leak the multi-hop route. We therefore apply entity-level masking over the expert search trace. For each trace, we

identify salient entity spans from answer spans, emphasized terms, and named entities detected by spaCy [15], and mask all occurrences within the same trace. This preserves the multi-step search trajectory while hiding entities that would otherwise reveal the search path or final answer.

Beyond these domains. This principle naturally extends beyond our evaluated domains. The three domains studied here cover common RLVR regimes: long-form reasoning with closed-form answer verification, execution-based generation, and multi-round tool interaction. Many other RLVR tasks can be viewed as variants or combinations of these regimes, with different semantic content lying on the critical path to the answer or task completion. Applying SMEPO to a new domain therefore amounts to identifying this domain-specific content and masking it while preserving the surrounding procedural guidance. For example, interactive agents extend multi-round tool interaction to web navigation, mobile-use, and embodied-control tasks, where SMEPO could mask URLs, page titles, object names, UI labels, coordinates, or action arguments while preserving the high-level action plan. Thus, what changes across settings is the domain-specific semantic type along the critical path, not the masked-guidance principle itself.

In the three domains studied here, SMEPO masking is applied offline with lightweight preprocessing. Across *all training data* for the three domains combined, preprocessing takes only *37 seconds in total* on the CPU node. This one-time cost is negligible compared with rollout generation and model optimization; detailed per-domain measurements are reported in Appendix C.

Overall, SMEPO differs from coarse trace truncation by preserving the trace whenever possible and masking only spans selected by reward-relevant rules. Rather than discarding large portions of the expert trace, SMEPO keeps the surrounding procedural guidance intact while removing verifier-aligned content along the critical path. We test this design choice in Section 3.4 by comparing against fixed-prefix exposure and same-budget random masking baselines.

3 Experiments

Models. We evaluate three open-weight models: Qwen3-8B-Base [17], Qwen2.5-7B [18], and DeepSeek-R1-Distill-Qwen-7B [2, 18]. Qwen3-8B-Base is our main model for math, code, and agentic search. Qwen2.5-7B and DeepSeek-R1-Distill-Qwen-7B are used as additional math models to test whether the degradation from full expert traces and the gains from SMEPO persist across different model families and base reasoning capabilities.

Evaluation tasks. We evaluate across math, code, and agentic search. For math, we use GSM8K [19], MATH500 [20], AIME25 [21], AIME26 [22], AMC [23], and OlympiadBench [24], and report the unweighted average over the six benchmarks. For AIME25 and AIME26, we average accuracy over 32 sampled responses per problem to reduce variance. For code, we evaluate pass@1 on HumanEval [25], HumanEval+ [26], and LiveCodeBench [27]. We also include a commonly used 131-problem LiveCodeBench v6 subset evaluation in Appendix A.1, where the same qualitative trend holds. For agentic search, we evaluate accuracy on Bamboogle [28]. We also report 5-shot MMLU-Overall [29, 30] in Appendix A.3 as a general-domain sanity check.

Training data and expert traces. The main math and code experiments use separate 1k-example subsets from OpenThoughts-114k [31]. Both subsets contain DeepSeek-R1 expert traces. In the main math experiments with Qwen3-8B-Base, we additionally evaluate Gemini-3-Pro-Preview traces to test robustness to the expert source. For agentic search, we use a 1k-example subset of MiroVerse HotpotQA search trajectories [32]. Within each experimental block, all methods use the same training problems, verifier, rollout budget, and expert source. Appendix A.2 reports an additional large-scale math experiment using 89k math examples from OpenThoughts.

Compared methods. We compare vanilla **GRPO** without expert traces; **SFT** on expert responses; and **SFT + RL**, which applies GRPO after SFT. For expert-trace guidance, **Expert** appends the full trace during GRPO, while **SMEPO** uses the same trace with reward-relevant spans masked. We also include **LUFFY** [12] and **GHPO** [9], both using the same expert traces as Expert and SMEPO.

Training and evaluation details. Unless otherwise noted, RL training uses 8 rollouts per prompt and runs for 400 optimization steps on 8×H100 GPUs. We use a maximum response length of 12k for math and code. For search, which is multi-round, we use a maximum response length of 2048 tokens

Math							
Method	GSM8K	MATH500	AIME25	AIME26	AMC	Olympiad	Avg.
Qwen3-8B-Base							
Qwen3-8B-Base	68.0	67.0	9.9	7.6	50.0	25.8	38.1
GRPO	91.3	75.2	12.5	12.3	65.0	29.5	47.6
SFT (DS)	88.0	71.2	8.0	5.6	50.0	25.4	41.4
SFT (DS) + RL	91.7	76.8	15.0	10.4	62.5	29.7	47.7
Expert (DS)	76.4	69.2	11.4	8.6	52.5	27.7	41.0
LUFFY (DS)	90.4	74.6	14.4	12.6	60.0	29.2	46.9
GHPO (DS)	79.5	73.6	13.5	10.0	60.0	29.2	44.3
SMEPO (DS)	92.2	76.6	**17.6**	13.4	**70.0**	32.0	**50.3**
SFT (Gemini)	91.7	71.2	10.2	9.0	52.5	25.2	43.3
SFT (Gemini) + RL	91.6	73.0	13.2	10.6	60.0	28.0	46.1
Expert (Gemini)	78.0	68.8	11.6	9.6	50.0	26.6	40.8
LUFFY (Gemini)	**92.4**	74.6	15.2	**14.7**	65.0	30.7	48.8
GHPO (Gemini)	79.4	73.2	13.9	9.5	65.0	29.5	45.1
SMEPO (Gemini)	92.3	**78.6**	16.1	14.4	67.5	**32.8**	**50.3**
Qwen2.5-7B							
Qwen2.5-7B	46.2	55.8	4.2	5.4	35.0	20.0	27.8
GRPO	87.6	70.2	9.4	7.7	55.0	24.3	42.4
SFT (DS)	87.0	66.8	7.3	5.0	42.5	22.1	38.5
Expert (DS)	85.3	63.8	5.6	6.4	47.5	22.8	38.6
SMEPO (DS)	**89.0**	**72.4**	**10.8**	**9.7**	**60.0**	**25.1**	**44.5**
DeepSeek-R1-Distill-Qwen-7B							
R1-Distill-Qwen-7B	83.5	81.0	32.6	33.3	82.5	35.3	58.0
GRPO	86.4	84.2	34.0	38.8	**90.0**	37.2	61.8
Expert (DS)	87.3	81.0	34.1	38.4	85.0	36.8	60.4
SMEPO (DS)	**87.5**	**85.4**	**34.7**	**39.7**	**90.0**	**38.3**	**62.6**

Code: Qwen3-8B-Base				
Method	HumanEval	HumanEval+	LCB	Avg.
Qwen3-8B-Base	63.4	55.5	35.0	51.3
GRPO	67.7	59.8	42.0	56.5
SFT	67.1	60.4	28.8	52.1
Expert	66.5	58.5	42.0	55.7
SMEPO	**69.5**	**62.2**	**45.3**	**59.0**

Agent: Qwen3-8B-Base	
Method	Bamboogle
Qwen3-8B-Base	20.8
GRPO	45.6
SFT	44.0
Expert	23.2
SMEPO	**48.8**

Table 1: **Main results**. (**top**) Math performance on Qwen3-8B-Base, Qwen2.5-7B, and DeepSeek-R1-Distill-Qwen-7B. The suffixes (DS) and (Gemini) indicate that the expert traces are sourced from DeepSeek-R1 and Gemini-3-Pro, respectively. (**bottom**) Code and agentic search performance on Qwen3-8B-Base. Across settings, naive full expert traces hurt or fail to help, while SMEPO consistently improves over both vanilla GRPO and naive expert-trace guidance. Best results are in bold; second-best results are underlined.

per round. The learning rate is 1×10^{-6} for math and code, and 8×10^{-7} for search. Expert traces or masked traces are appended after the original problem as additional context and are used only during rollout generation. Rewards are computed from the model-generated answer using the same verifier as vanilla GRPO. Evaluation uses lm-evaluation-harness [33] for supported benchmarks; for benchmarks not supported by lm-evaluation-harness, evaluation details are included in Appendix E.1.

3.1 Main results

Table 1 shows that complete expert traces consistently fail to improve vanilla GRPO, while masked expert traces improve performance across domains. In the main math setting with Qwen3-8B-Base, vanilla GRPO achieves a six-benchmark average of 47.6. Appending complete expert traces reduces the average to 41.0 with DeepSeek-R1 traces and 40.8 with Gemini traces. In contrast,



(a) LiveCodeBench curve.

Model	Method	Step	Time (hr)
Qwen3-8B-Base	GRPO	255	9.90
	SMEPO	55	2.37
Qwen2.5-7B	GRPO	300	8.00
	SMEPO	175	4.57
R1-Distill-Qwen-7B	GRPO	350	14.56
	SMEPO	270	10.99

(b) Steps and wall-clock time to target.

Figure 3: **Training dynamics and target-reaching efficiency.** (Left): LiveCodeBench curve for Qwen3-8B-Base with DeepSeek-R1 traces. (Right): time to reach the GRPO peak mean@32 accuracy (averaged over AIME25 and AIME26).

SMEPO reaches 50.3 with both expert sources, improving over vanilla GRPO by 2.7 points and over full-expert-trace conditioning by 9.3–9.5 points.

The same trend holds across other models. On Qwen2.5-7B, full-expert-trace conditioning reduces the average from 42.4 to 38.6, while SMEPO improves it to 44.5. On DeepSeek-R1-Distill-Qwen-7B, full-expert-trace conditioning reduces the average from 61.8 to 60.4, while SMEPO improves it to 62.6. The smaller degradation on DeepSeek-R1-Distill-Qwen-7B likely reflects its stronger prior reasoning ability, which makes it less dependent on exposed expert content than base models.

For code, complete expert traces are less harmful than in math and agentic search, likely because exploiting them requires reproducing or adapting a longer executable solution rather than copying short reward-relevant content such as numbers or entities. On Qwen3-8B-Base, GRPO obtains a three-benchmark average of 56.5, full-expert-trace conditioning obtains 55.7, and SMEPO improves the average to 59.0. SMEPO also gives the best score on each reported code benchmark. Although the accuracy drop from full traces is smaller in code than in math or search, Figure 4 shows that full-expert-trace conditioning substantially increases copying from the visible expert solution and reuse of the original expert program, whereas SMEPO suppresses both effects. This indicates that complete code traces still bias the policy toward trace reuse, while SMEPO reduces this dependency by masking the executable solution span.

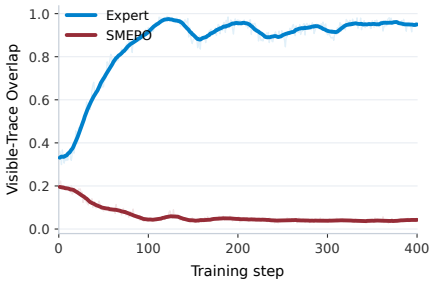
For agentic search, the degradation from complete expert traces is the largest. On Bamboogle, GRPO reaches 45.6, full-expert-trace conditioning drops to 23.2, and SMEPO improves to 48.8. This sharp drop suggests that full traces can suppress the policy's active use of the search format: instead of issuing proper search calls, the model relies on exposed trajectory content. Since this tool-use behavior is not strongly present in the base model, the policy has less prior capability to fall back on once active search is weakened.

Overall, these results suggest that the failure of full-expert-trace conditioning is not specific to one model, one expert source, or one domain, and that masking expert traces provides a more reliable way to use expert guidance in RLVR.

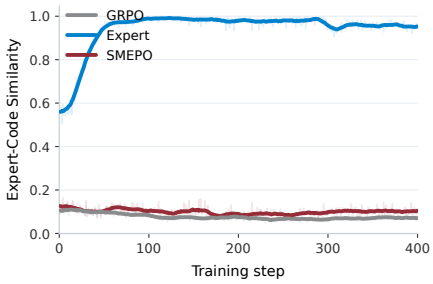
3.2 In-depth analysis beyond accuracy

Training efficiency. Figure 3(b) reports the time required to reach a fixed hard-math target. SMEPO reaches this target significantly faster than vanilla GRPO across all three models. For instance, on Qwen3-8B-Base, it requires only 2.37 hours compared to GRPO's 9.90 hours. Consequently, SMEPO enhances not only the final accuracy but also the convergence efficiency.

Analysis of copying behavior. Figure 4 illustrates two copying patterns. Unlike full-expert conditioning, which yields high *visible-trace overlap* and *expert-code similarity*, SMEPO maintains near-zero overlap by masking executable solution spans. This demonstrates that complete code traces create an unintended reward channel, where policies gain rewards by simply reproducing prompt-exposed content. By suppressing this channel, SMEPO ensures the model utilizes high-level guidance—such as algorithmic plans and task interpretation—rather than merely adapting expert code. See Appendix B.1 for detailed metric definitions.



(a) Visible-trace overlap.



(b) Expert-code similarity.

Figure 4: **Code-copy diagnostics. Left:** *Visible-trace overlap* captures copying from the expert content actually visible in the prompt. **Right:** *Expert-code similarity* captures reuse of the original expert program. The left figure does not include GRPO since it does not involve expert traces.

Method	GSM8K	MATH500	AIME25	AIME26	AMC	Olympiad	Avg.
Qwen3-8B-Base	68.0	67.0	9.9	7.6	50.0	25.8	38.1
GHPO	78.3	72.8	13.4	**10.2**	60.0	**29.5**	44.0
GHPO + SMEPO	**82.1**	**74.2**	**15.1**	10.1	**62.5**	**29.5**	**45.6**

Method	HumanEval	HumanEval+	LCB	Avg.
Qwen3-8B-Base	63.4	55.5	35.0	51.3
GHPO	65.8	**58.5**	40.3	54.9
GHPO + SMEPO	**67.1**	57.9	**41.8**	**55.6**

Method	Bamboogle
Qwen3-8B-Base	20.8
GHPO	42.4
GHPO + SMEPO	**44.0**

Table 2: **Additional gains with GHPO.** On math benchmarks with Qwen3-8B-Base, adding SMEPO to GHPO improves the six-benchmark average from 44.0 to 45.6, suggesting that masked expert traces can bring additional gains on top of prompt-refinement guidance. Similar improvements are observed in code (54.9 to 55.6) and agentic search (42.4 to 44.0), using adapted GHPO implementations. Best results are in bold; second-best results are underlined.

3.3 Compatibility with other guidance methods

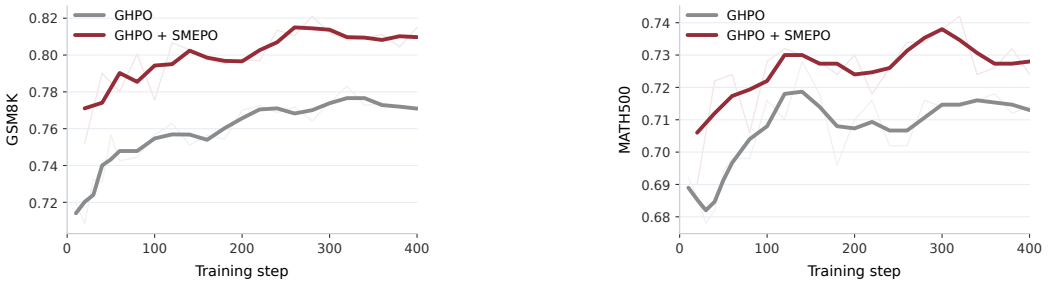
Table 2 evaluates SMEPO integrated with GHPO using identical training configurations within each setting. Adding SMEPO masking improves Qwen3-8B-Base performance from 44.0 to 45.6 on math, 54.9 to 55.6 on code, and 42.4 to 44.0 on agentic search. For code and agentic search, we adapt GHPO to the corresponding task settings. As shown in Figure 5, these gains are additive rather than redundant, suggesting masked traces provide unique benefits beyond GHPO's original signals.

3.4 Ablations

Table 3 studies which part of the expert-trace transformation is responsible for the gains. First, prefix exposure shows that simply revealing less of the trace is insufficient. Using 75%, 50%, or 25% of the expert prefix improves over full-expert-trace conditioning, but the gains are non-monotonic and remain below SMEPO. This suggests that the issue is not only the length or amount of expert text, but whether reward-relevant spans remain visible.

Second, random masking performs poorly. Random word masking and random sentence masking both underperform full-expert-trace conditioning and are far below SMEPO. Both random baselines use a masking budget matched to SMEPO for each problem. Thus, the benefit of SMEPO is not due to the amount of removed text, generic noise injection, or prompt corruption.

Third, reward-aware masking shows a clear progression. Masking only the final answer improves over full-expert-trace conditioning; masking all answer occurrences improves further; and SMEPO performs best by masking numeric content throughout the math trace. This supports our diagnosis that reward-relevant information is distributed across expert traces: final answers, repeated answer values, and intermediate numeric values can all act as exploitable reward channels. SMEPO works best because it removes these channels while preserving the surrounding procedural guidance.



(a) GSM8K. (b) MATH500.

Figure 5: **Training curves with GHPO.** Using the original GHPO setup, adding SMEPO improves the model accuracy on GSM8K and MATH500.

Cat.	Method	GSM8K	MATH500	AIME25	AIME26	AMC	Olympiad	Avg.
Full	Expert (100%)	76.4	69.2	11.4	8.6	52.5	27.7	41.0
Prefix	Expert (75%)	86.1	75.0	<u>14.5</u>	11.6	60.0	29.7	46.1
	Expert (50%)	85.2	72.6	13.5	10.3	57.5	28.8	44.7
	Expert (25%)	87.1	74.6	14.2	11.0	<u>65.0</u>	<u>30.6</u>	47.1
Random	Word	73.6	67.0	10.2	8.4	50.0	25.2	39.1
	Sentence	75.7	68.0	10.9	9.4	45.0	26.9	39.3
Targeted	Mask Final	82.9	70.4	11.7	9.1	57.5	27.9	43.2
	Mask Ans. Uses	<u>89.9</u>	<u>75.2</u>	14.4	<u>13.2</u>	<u>65.0</u>	30.4	<u>48.0</u>
	SMEPO	**92.2**	**76.6**	**17.6**	**13.4**	**70.0**	**32.0**	**50.3**

Table 3: **Math ablations with Qwen3-8B-Base.** We compare full-expert-trace, prefix exposure, random masking, and targeted masking. Best results are in bold; second-best results are underlined.

4 Related Work

RLVR and guided reasoning. RLVR improves reasoning models with verifier-based feedback in math and code [1–5, 34], and has been extended to agentic search and research environments with tool-use and retrieval trajectories, such as Search-R1 [35] and DeepResearcher [7]. Recent work studies how to scale RLVR, stabilize critic-free optimization, and improve exploration under sparse rewards [4–6]. Closely related methods use off-policy or privileged guidance: LUFFY [12] incorporates off-policy reasoning traces during RL; Guide [13] injects hints when all initial rollouts fail; GHPO [9] provides difficulty-aware prompt refinement; TRAPO [10] interleaves expert-prefix supervision with RL; POPE [11] uses oracle prefixes for hard problems; and Learning-like-humans [36] uses expert traces to reformulate hard problems. These methods mainly regulate *when* or *how much* expert signal is shown, while we ask *which parts* of an expert trace should be shown.

Reward hacking under expert guidance. Reward hacking occurs when a learner obtains high reward through unintended strategies rather than the intended behavior [37–39]. We study a guidance-induced form, where expert traces expose reward-relevant values, executable programs, or entities along the critical path that the policy can exploit during RL.

Masked and fill-in-the-blank-style learning. Masking and infilling learn from partially observed context: BERT [40], MASS [41], BART [42], and T5 [43] reconstruct masked or corrupted text, while fill-in-the-middle training [44] conditions on prefix and suffix context. Code models also use FIM or fill-in-the-blank objectives, including InCoder [45], SantaCoder [46], CodeT5+ [47], DeepSeek-Coder [48], and DeepSeek-Coder-V2 [49], with syntax- or structure-aware variants such as SAFIM [50] and AST-FIM [51]. SMEPO brings a related masking principle into expert-guided RL, using masked traces as exploration guidance rather than supervised reconstruction targets.

5 Conclusion

Expert traces provide useful guidance but can also expose reward-informative content, including final answers and critical-path intermediate information. SMEPO masks these spans while preserving procedural cues, turning expert traces into fill-in-the-blank-style guidance for RLVR. Across math, code, and agentic search, SMEPO improves over vanilla GRPO and full-expert-trace conditioning, suggesting that effective expert-guided RL should preserve useful guidance while masking reward-relevant content along the critical path, rather than exposing expert traces unchanged.

Acknowledgments and Disclosure of Funding

Funding. We thank MIT-IBM Watson AI Lab, MIT AI Hardware Program, MIT Amazon Science Hub, Hyundai Motor Company, and National Science Foundation for supporting this work.

Competing Interests. The authors declare no competing interests.

References

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

[2] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Honghui Ding, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jingchang Chen, Jingyang Yuan, Jinhao Tu, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaichao You, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingxu Zhou, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645(8081):633–638, September 2025. ISSN 1476-4687. doi: 10.1038/s41586-025-09422-z. URL http://dx.doi.org/10.1038/s41586-025-09422-z.

[3] Jian Hu, Jason Klein Liu, Haotian Xu, and Wei Shen. REINFORCE++: Stabilizing critic-free policy optimization with global advantage normalization, 2025. URL https://arxiv.org/abs/2501.03262.

[4] Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. DAPO: An open-source LLM reinforcement learning system at scale, 2025. URL https://arxiv.org/abs/2503.14476.

[5] Yiping Wang, Qing Yang, Zhiyuan Zeng, Liliang Ren, Liyuan Liu, Baolin Peng, Hao Cheng, Xuehai He, Kuan Wang, Jianfeng Gao, Weizhu Chen, Shuohang Wang, Simon Shaolei Du, and Yelong Shen. Reinforcement learning for reasoning in large language models with one training example, 2025. URL https://arxiv.org/abs/2504.20571.

[6] Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding R1-Zero-Like training: A critical perspective, 2025. URL https://arxiv.org/abs/2503.20783.

[7] Yuxiang Zheng, Dayuan Fu, Xiangkun Hu, Xiaojie Cai, Lyumanshan Ye, Pengrui Lu, and Pengfei Liu. DeepResearcher: Scaling deep research via reinforcement learning in real-world environments, 2025. URL https://arxiv.org/abs/2504.03160.

[8] Yangzhen Wu, Shanda Li, Zixin Wen, Xin Zhou, Ameet Talwalkar, Yiming Yang, Wenhao Huang, and Tianle Cai. Learn hard problems during RL with reference guided fine-tuning, 2026. URL https://arxiv.org/abs/2603.01223.

[9] Ziru Liu, Cheng Gong, Xinyu Fu, Yaofang Liu, Ran Chen, Shoubo Hu, Suiyun Zhang, Rui Liu, Qingfu Zhang, and Dandan Tu. GHPO: Adaptive guidance for stable and efficient LLM reinforcement learning, 2025. URL https://arxiv.org/abs/2507.10628.

[10] Mingyu Su, Jian Guan, Yuxian Gu, Minlie Huang, and Hongning Wang. Trust-region adaptive policy optimization, 2025. URL https://arxiv.org/abs/2512.17636.

[11] Yuxiao Qu, Amrith Setlur, Virginia Smith, Ruslan Salakhutdinov, and Aviral Kumar. POPE: Learning to reason on hard problems via privileged on-policy exploration, 2026. URL https://arxiv.org/abs/2601.18779.

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

[13] Vaskar Nath, Elaine Lau, Anisha Gunjal, Manasi Sharma, Nikhil Baharte, and Sean Hendryx. Adaptive guidance accelerates reinforcement learning of reasoning models, 2025. URL https://arxiv.org/abs/2506.13923.

[14] Trung Quoc Luong, Xinbo Zhang, Zhanming Jie, Peng Sun, Xiaoran Jin, and Hang Li. ReFT: Reasoning with reinforced fine-tuning, 2024. URL https://arxiv.org/abs/2401.08967.

[15] Ines Montani, Matthew Honnibal, Adriane Boyd, Sofie Van Landeghem, and Henning Peters. explosion/spaCy: v3.7.2: Fixes for APIs and requirements, October 2023. URL https://doi.org/10.5281/zenodo.10009823.

[16] Urchade Zaratiana, Nadi Tomeh, Pierre Holat, and Thierry Charnois. GLiNER: Generalist model for named entity recognition using bidirectional transformer. In Kevin Duh, Helena Gomez, and Steven Bethard, editors, *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 5364–5376, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.300. URL https://aclanthology.org/2024.naacl-long.300/.

[17] 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, 2025. URL https://arxiv.org/abs/2505.09388.

[18] Qwen Team, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

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

[20] Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step, 2023. URL https://arxiv.org/abs/2305.20050.

[21] Yifan Zhang and Team Math-AI. American invitational mathematics examination (AIME) 2025, 2025. URL https://huggingface.co/datasets/math-ai/aime25.

[22] Yifan Zhang and Team Math-AI. American invitational mathematics examination (AIME) 2026, 2026. URL https://huggingface.co/datasets/math-ai/aime26.

[23] Yifan Zhang and Team Math-AI. American mathematics competitions (AMC) 2023, 2023. URL https://huggingface.co/datasets/math-ai/amc23.

[24] Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. OlympiadBench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems, 2024. URL https://arxiv.org/abs/2402.14008.

[25] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021. URL https://arxiv.org/abs/2107.03374.

[26] Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by ChatGPT really correct? rigorous evaluation of large language models for code generation, 2023. URL https://arxiv.org/abs/2305.01210.

[27] Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. LiveCodeBench: Holistic and contamination free evaluation of large language models for code, 2024. URL https://arxiv.org/abs/2403.07974.

[28] Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A. Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models, 2023. URL https://arxiv.org/abs/2210.03350.

[29] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

[30] Dan Hendrycks, Collin Burns, Steven Basart, Andrew Critch, Jerry Li, Dawn Song, and Jacob Steinhardt. Aligning ai with shared human values. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

[31] Etash Guha, Ryan Marten, Sedrick Keh, Negin Raoof, Georgios Smyrnis, Hritik Bansal, Marianna Nezhurina, Jean Mercat, Trung Vu, Zayne Sprague, Ashima Suvarna, Benjamin Feuer, Liangyu Chen, Zaid Khan, Eric Frankel, Sachin Grover, Caroline Choi, Niklas Muennighoff, Shiye Su, Wanjia Zhao, John Yang, Shreyas Pimpalgaonkar, Kartik Sharma, Charlie Cheng-Jie Ji, Yichuan Deng, Sarah Pratt, Vivek Ramanujan, Jon Saad-Falcon, Jeffrey Li, Achal Dave, Alon Albalak, Kushal Arora, Blake Wulfe, Chinmay Hegde, Greg Durrett, Sewoong Oh, Mohit Bansal, Saadia Gabriel, Aditya Grover, Kai-Wei Chang, Vaishaal Shankar, Aaron Gokaslan, Mike A. Merrill, Tatsunori Hashimoto, Yejin Choi, Jenia Jitsev, Reinhard Heckel, Maheswaran Sathiamoorthy, Alexandros G. Dimakis, and Ludwig Schmidt. OpenThoughts: Data recipes for reasoning models, 2025. URL https://arxiv.org/abs/2506.04178.

[32] MiroMind Team, Song Bai, Lidong Bing, Carson Chen, Guanzheng Chen, Yuntao Chen, Zhe Chen, Ziyi Chen, Jifeng Dai, Xuan Dong, Wenhan Dou, Yue Deng, Yunjie Fu, Junqi Ge, Chenxia Han, Tammy Huang, Zhenhang Huang, Jerry Jiao, Shilei Jiang, Tianyu Jiao, Xiaoqi Jian, Lei Lei, Ruilin Li, Gen Luo, Tiantong Li, Xiang Lin, Ziyuan Liu, Zhiqi Li, Jie Ni, Qiang Ren, Pax Sun, Shiqian Su, Chenxin Tao, Bin Wang, Wenhai Wang, Haonan Wang, James Wang, Jin Wang, Jojo Wang, Letian Wang, Shizun Wang, Weizhi Wang, Zixuan Wang, Jinfan Xu, Sen Xing, Chenyu Yang, Hai Ye, Jiaheng Yu, Yue Yu, Muyan Zhong, Tianchen Zhao, Xizhou Zhu, Yanpeng Zhou, Yifan Zhang, and Zhi Zhu. MiroThinker: Pushing the performance boundaries of open-source research agents via model, context, and interactive scaling, 2026. URL https://arxiv.org/abs/2511.11793.

[33] Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, July 2024. URL https://zenodo.org/records/12608602.

[34] Ruitao Liu, Qinghao Hu, and Song Han. d-opd: Future-aware on-policy distillation for block diffusion language models, 2026. URL https://arxiv.org/abs/2609.35362.

[35] Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-R1: Training LLMs to reason and leverage search engines with reinforcement learning, 2025. URL https://arxiv.org/abs/2503.09516.

[36] Enci Zhang, Xingang Yan, Wei Lin, Tianxiang Zhang, and Qianchun Lu. Learning Like Humans: Advancing LLM reasoning capabilities via adaptive difficulty curriculum learning and expert-guided self-reformulation, 2025. URL https://arxiv.org/abs/2505.08364.

[37] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in AI safety, 2016. URL https://arxiv.org/abs/1606.06565.

[38] Tom Everitt, Marcus Hutter, Ramana Kumar, and Victoria Krakovna. Reward tampering problems and solutions in reinforcement learning: A causal influence diagram perspective, 2021. URL https://arxiv.org/abs/1908.04734.

[39] Joar Skalse, Nikolaus Howe, Dmitrii Krasheninnikov, and David Krueger. Defining and characterizing reward gaming. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 9460–9471. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/3d719fee332caa23d5038b8a90e81796-Paper-Conference.pdf.

[40] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423/.

[41] Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. MASS: Masked sequence to sequence pre-training for language generation, 2019. URL https://arxiv.org/abs/1905.02450.

[42] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.703. URL https://aclanthology.org/2020.acl-main.703/.

[43] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. URL http://jmlr.org/papers/v21/20-074.html.

[44] Mohammad Bavarian, Heewoo Jun, Nikolas Tezak, John Schulman, Christine McLeavey, Jerry Tworek, and Mark Chen. Efficient training of language models to fill in the middle, 2022. URL https://arxiv.org/abs/2207.14255.

[45] Daniel Fried, Armen Aghajanyan, Jessy Lin, Sida Wang, Eric Wallace, Freda Shi, Ruiqi Zhong, Wen tau Yih, Luke Zettlemoyer, and Mike Lewis. InCoder: A generative model for code infilling and synthesis, 2023. URL https://arxiv.org/abs/2204.05999.

[46] Loubna Ben Allal, Raymond Li, Denis Kocetkov, Chenghao Mou, Christopher Akiki, Carlos Munoz Ferrandis, Niklas Muennighoff, Mayank Mishra, Alex Gu, Manan Dey, Logesh Kumar Umapathi, Carolyn Jane Anderson, Yangtian Zi, Joel Lamy Poirier, Hailey Schoelkopf, Sergey Troshin, Dmitry Abulkhanov, Manuel Romero, Michael Lappert, Francesco De Toni, Bernardo García del Río, Qian Liu, Shamik Bose, Urvashi Bhattacharyya, Terry Yue Zhuo, Ian Yu, Paulo Villegas, Marco Zocca, Sourab Mangrulkar, David Lansky, Huu Nguyen, Danish Contractor, Luis Villa, Jia Li, Dzmitry Bahdanau, Yacine Jernite, Sean Hughes, Daniel Fried, Arjun Guha, Harm de Vries, and Leandro von Werra. SantaCoder: don't reach for the stars!, 2023. URL https://arxiv.org/abs/2301.03988.

[47] Yue Wang, Hung Le, Akhilesh Deepak Gotmare, Nghi D. Q. Bui, Junnan Li, and Steven C. H. Hoi. CodeT5+: Open code large language models for code understanding and generation, 2023. URL https://arxiv.org/abs/2305.07922.

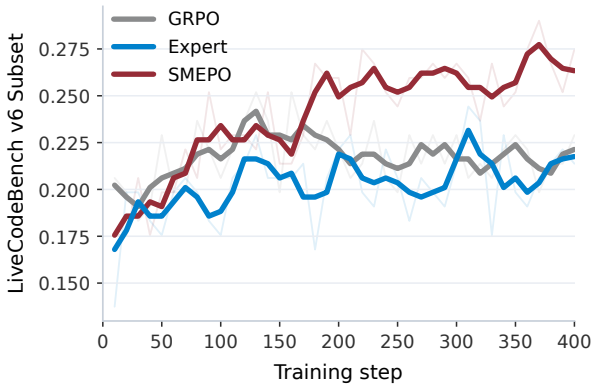
[48] Daya Guo, Qihao Zhu, Dejian Yang, Zhenda Xie, Kai Dong, Wentao Zhang, Guanting Chen, Xiao Bi, Y. Wu, Y. K. Li, Fuli Luo, Yingfei Xiong, and Wenfeng Liang. DeepSeek-Coder: When the large language model meets programming – the rise of code intelligence, 2024. URL https://arxiv.org/abs/2401.14196.

[49] DeepSeek-AI, Qihao Zhu, Daya Guo, Zhihong Shao, Dejian Yang, Peiyi Wang, Runxin Xu, Y. Wu, Yukun Li, Huazuo Gao, Shirong Ma, Wangding Zeng, Xiao Bi, Zihui Gu, Hanwei Xu, Damai Dai, Kai Dong, Liyue Zhang, Yishi Piao, Zhibin Gou, Zhenda Xie, Zhewen Hao, Bingxuan Wang, Junxiao Song, Deli Chen, Xin Xie, Kang Guan, Yuxiang You, Aixin Liu, Qiushi Du, Wenjun Gao, Xuan Lu, Qinyu Chen, Yaohui Wang, Chengqi Deng, Jiashi Li, Chenggang Zhao, Chong Ruan, Fuli Luo, and Wenfeng Liang. DeepSeek-Coder-V2: Breaking the barrier of closed-source models in code intelligence, 2024. URL https://arxiv.org/abs/2406.11931.

[50] Linyuan Gong, Sida Wang, Mostafa Elhoushi, and Alvin Cheung. Evaluation of LLMs on syntax-aware code fill-in-the-middle tasks, 2024. URL https://arxiv.org/abs/2403.04814.

[51] Linyuan Gong, Alvin Cheung, Mostafa Elhoushi, and Sida Wang. Structure-aware fill-in-the-middle pretraining for code, 2025. URL https://arxiv.org/abs/2506.00204.

[52] Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. HybridFlow: A flexible and efficient RLHF framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, EuroSys '25, page 1279–1297. ACM, March 2025. doi: 10.1145/3689031.3696075. URL http://dx.doi.org/10.1145/3689031.3696075.



LCB v6 Subset	
Method	Pass@1
Qwen3-8B-Base	14.5
GRPO	25.2
Expert	24.4
SMEPO	**29.0**

Figure 6: **Additional LiveCodeBench v6 subset evaluation.** Left: Training dynamics; right: final pass@1 scores. We evaluate Qwen3-8B-Base on 131 problems released between 2025-02-01 and 2025-05-01. SMEPO performs best, consistent with the main code evaluation.

A Additional Results

A.1 Additional LiveCodeBench v6 subset evaluation

We further evaluate Qwen3-8B-Base on a commonly used LiveCodeBench v6 subset, consisting of 131 problems released between 2025-02-01 and 2025-05-01. This evaluation complements the LiveCodeBench result in the main code table and provides an additional check on a recent code-generation subset used by the community.

Figure 6 reports both the training dynamics and final pass@1 scores on this subset. SMEPO achieves the best final performance, improving over GRPO from 25.2 to 29.0 and over full expert-trace conditioning from 24.4 to 29.0. The result is consistent with our main code evaluation: directly exposing complete expert code can hurt performance relative to vanilla GRPO, while masking executable solution content yields better performance.

A.2 Large-scale 89k math pool

We additionally evaluate SMEPO on a larger 89k-example math training pool with Qwen3-8B-Base to test whether the same trend holds beyond the curated 1k hard-reasoning setting.

Table 4 shows a consistent pattern. Vanilla GRPO improves the six-benchmark average from 38.1 to 47.8, while direct expert conditioning with DeepSeek-R1 traces decreases the average to 36.6. In contrast, SMEPO achieves 49.4, outperforming both vanilla GRPO and full-expert-trace conditioning. This result suggests that the benefit of masking is not limited to the small curated training set and remains effective at a larger data scale.

A.3 General-domain checks on MMLU

We report MMLU-Overall as a general-domain sanity check rather than a main target metric. Since SMEPO modifies the guidance context used during GRPO, one concern is that the gains on math, code, or agentic search might come at the cost of broader model capability. MMLU is not designed to measure the specific reasoning behaviors targeted by SMEPO, but it provides a useful check of whether training causes a broad regression outside the target tasks.

Table 5 shows that SMEPO remains essentially flat relative to the base model and vanilla GRPO across the math, code, and agentic-search settings. In several cases, SMEPO slightly improves MMLU-Overall, while SFT more visibly lowers MMLU. This is consistent with the role of SMEPO as a guidance transformation rather than pure imitation: it changes what expert context the policy sees during GRPO, but it still optimizes the same verifier-based objective. Overall, the MMLU results suggest that the task gains from masked expert guidance do not come from sacrificing broad general-domain performance.

Method	GSM8K	MATH500	AIME25	AIME26	AMC	Olympiad	Avg.
Qwen3-8B-Base	68.0	67.0	9.9	7.6	50.0	25.8	38.1
GRPO	<u>92.3</u>	<u>77.4</u>	<u>13.0</u>	<u>9.3</u>	<u>65.0</u>	<u>30.1</u>	<u>47.8</u>
Expert (DS)	75.7	63.2	9.0	6.9	42.5	22.7	36.6
SMEPO (DS)	**93.6**	**77.6**	**14.7**	**11.4**	**67.5**	**31.5**	**49.4**

Table 4: **Large-scale 89k-example math results with Qwen3-8B-Base.** The same pattern persists at scale: full expert-trace conditioning underperforms vanilla GRPO, while SMEPO improves beyond both. Best results are in bold; second-best results are underlined.

Qwen3-8B-Base Math

Method	MMLU
Qwen3-8B-Base	75.7
GRPO	75.7
SFT (DS)	74.5
SFT (DS) + RL	74.8
Expert (DS)	<u>75.8</u>
LUFFY (DS)	75.6
GHPO (DS)	75.0
SMEPO (DS)	75.7
SFT (Gemini)	74.1
SFT (Gemini) + RL	74.9
Expert (Gemini)	75.5
LUFFY (Gemini)	75.6
GHPO (Gemini)	75.3
SMEPO (Gemini)	**75.9**

Qwen3-8B-Base Code

Method	MMLU
Qwen3-8B-Base	<u>75.7</u>
GRPO	<u>75.7</u>
SFT	74.4
Expert	**75.8**
SMEPO	**75.8**

Qwen3-8B-Base Agent

Method	MMLU
Qwen3-8B-Base	<u>75.7</u>
GRPO	**75.8**
SFT	75.0
Expert	75.6
SMEPO	**75.8**

Qwen2.5-7B Math

Method	MMLU
Qwen2.5-7B	72.0
GRPO	72.2
SFT	71.8
Expert	<u>72.4</u>
SMEPO	**72.5**

R1-Distill-Qwen-7B Math

Method	MMLU
R1-Distill-7B	50.7
GRPO	50.6
Expert	<u>50.8</u>
SMEPO	**51.3**

Table 5: **General-domain sanity check.** MMLU-Overall results for the math, code, and agent runs. SMEPO preserves broad general-domain performance across settings and often stays flat or slightly improves, while SFT tends to cause a clearer drop. Best results are in bold; second-best results are underlined.

A.4 Diagnostic summary of full-expert-trace conditioning

Table 6 summarizes the effect of directly appending complete expert traces during GRPO. Across all reported settings, full-expert-trace conditioning underperforms vanilla GRPO, although the traces come from strong expert models such as DeepSeek-R1 or Gemini-3-Pro. This supports the central observation of the paper: complete expert traces are not simply helpful demonstrations, because they can expose content that is too directly tied to verifier reward.

The drop is largest in agentic search, where complete trajectories can expose answer-relevant entities and search pivots that suppress the model's active use of the search format. The degradation is smaller for code and for DeepSeek-R1-Distill-Qwen-7B, but the direction remains negative. This matches the analysis in the main text: code traces require reproducing or adapting a longer executable solution, and the reasoning-distilled model has stronger existing reasoning ability, so both settings are less vulnerable than base-model math or agentic search. Overall, this diagnostic table reinforces our interpretation that expert traces are useful only when their guidance is separated from directly reward-relevant content.

B Additional Analysis

The main text reports ablation results comparing different ways of exposing or masking expert traces. In this section, we provide additional analyses to better understand the behavior behind these ablations.

Model	Task	Expert trace	GRPO Avg.	Expert Avg.	Δ
Qwen3-8B-Base	Math	DeepSeek-R1	47.6	41.0	-6.6
Qwen3-8B-Base	Math	Gemini-3-Pro	47.6	40.8	-6.8
Qwen2.5-7B	Math	DeepSeek-R1	42.4	38.6	-3.8
R1-Distill-Qwen-7B	Math	DeepSeek-R1	61.8	60.4	-1.4
Qwen3-8B-Base	Code	DeepSeek-R1	56.5	55.7	-0.8
Qwen3-8B-Base	Agent	DeepSeek-R1	45.6	23.2	-22.4

Table 6: **Diagnostic summary of full-expert-trace conditioning.** All values are taken from Table 1. Avg. denotes the task-level average used in the main table: six math benchmarks for Math, three code benchmarks for Code, and Bamboogle accuracy for Agent. Δ is Expert Avg. minus GRPO Avg.; negative values indicate that full expert-trace conditioning underperforms vanilla GRPO.

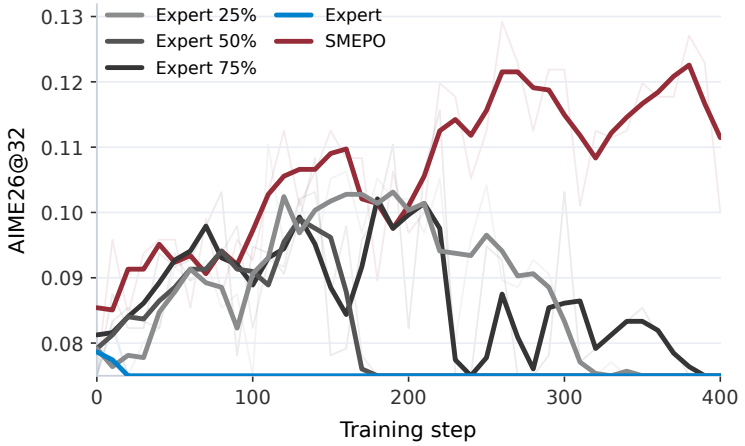


Figure 7: **Effect of partial expert-trace exposure on AIME26.** Coarse prefix exposure reduces the harm of full-expert-trace conditioning, but its effect is non-monotonic and remains below SMEPO, suggesting that the key issue is not trace length alone.

B.1 Code-copy diagnostics.

For code tasks, the reward-relevant content is usually a complete executable solution rather than a short numeric answer or entity. We therefore measure whether expert guidance encourages direct copying using two complementary metrics.

Let G_{rollout} denote the model rollout and G_{code} denote the generated code block, T_{vis} denote the portion of the expert trace that is visible in the prompt, and T_{code} denote the original expert code block. We define *Visible-trace overlap* as

$$\text{VisibleTraceOverlap} = \frac{\text{LCCS}(G_{\text{rollout}}, T_{\text{vis}})}{|G_{\text{rollout}}|},$$

where $\text{LCCS}(\cdot,\cdot)$ is the length of the longest common contiguous subsequence and $|G_{\text{rollout}}|$ is the rollout length. This metric captures whether the model copies content that was actually exposed in the prompt.

We further define *Expert-code similarity* as

$$\text{ExpertCodeSimilarity} = \frac{\text{LCCS}(G_{\text{code}}, T_{\text{code}})}{|G_{\text{code}}|}.$$

This metric measures similarity between the generated code block and the original expert code block, independent of whether that code block was fully visible or masked in the prompt.

B.2 Partial exposure.

Figure 7 left shows that exposing only a prefix of the expert trace is a useful but incomplete control. Prefix exposure reduces the amount of expert content visible to the policy, and therefore can reduce

the harm of full-expert-trace conditioning. However, it does not explicitly distinguish guidance from reward-relevant content. Since reward-relevant spans can appear at different positions across traces, a fixed prefix ratio may remove useful later reasoning steps while still leaving exploitable intermediate values or answer-related content visible. This explains the non-monotonic behavior of prefix exposure: shorter traces are not necessarily safer, and longer traces are not necessarily more useful. SMEPO instead keeps the trace organization whenever possible and masks spans selected by reward-relevant rules, which gives a more direct control over the harmful channel.

B.3 Why masking numbers outperforms answer-only masking.

The ablation results in the main text show that masking only the final answer is not sufficient. In math traces, the final answer is often only one instance of a broader value channel. Expert solutions may repeat the answer value, introduce intermediate quantities that nearly determine the answer, or include arithmetic correlations that make the hidden answer easy to recover. As a result, answer-only masking can still leave enough numeric evidence for the policy to exploit the trace without learning the underlying reasoning process. SMEPO masks numeric content throughout the expert trace, removing this broader reward-relevant payload while preserving equation templates, decomposition patterns, and surrounding explanatory text. This is why numeric masking outperforms answer-only masking: it targets the distributed value information that makes expert traces too easy to exploit.

C Method Overhead

We measure the offline preprocessing overhead of SMEPO on the same CPU node for all domains. The machine has dual Intel Xeon Gold 6442Y processors, with 48 physical cores and 96 logical CPUs. All measurements are performed without GPU acceleration. We use `/usr/bin/time -v` to measure the end-to-end wall-clock time of each preprocessing script, including reading the original data, applying the corresponding masking rule, and writing the processed output files.

For math, the preprocessing script masks numeric reward-relevant spans in expert reasoning traces. For code, the script masks executable code blocks in expert solutions while preserving the surrounding natural-language guidance. For agentic search, the script masks answer-relevant entities in teacher search rounds using rule-based answer/entity extraction and spaCy NER. These preprocessing steps are performed once offline before RL training, and the resulting masked data are reused throughout training.

Table 7 shows that SMEPO adds only a small offline preprocessing cost. Processing 1,024 examples takes 5.96 seconds for math, 1.80 seconds for code, and 29.42 seconds for agentic search. The search setting is slower because it performs entity-level masking with spaCy NER [15], whereas math and code use lightweight rule-based masking. Since masking is applied once before training, this cost is amortized over all RL updates and is negligible compared with rollout generation and model optimization.

D Case Studies

Because the original model outputs are in Markdown, we present the case studies in Markdown format without converting them into LaTeX or otherwise altering their content.

D.1 Annotated Case Summaries

We first present annotated summaries of three representative case studies. These summaries are drawn from the full rollouts in Appendix D.2 and highlight the most diagnostic parts of each example: what the unmasked expert hint exposes, how the Expert rollout responds, where the vanilla GRPO rollout fails, and how SMEPO uses the masked trace.

The summaries are not separate examples from the full cases below. Instead, they provide a compact, analysis-oriented view of the same rollouts before the complete transcripts are shown.

Domain	Examples	Wall-clock time	Time / example	Peak memory (MiB)
Math	1,024	5.96s	5.82ms	365
Code	1,024	1.80s	1.76ms	727
Agentic search	1,024	29.42s	28.7ms	850

Table 7: **Offline preprocessing overhead.** We measure end-to-end preprocessing time with `/usr/bin/time -v` on the same CPU node for all domains. Timing includes script startup, reading the original data, applying SMEPO masking, validation checks when present, and writing the processed JSONL/parquet outputs.

Annotated Summary: Math Reasoning with Expert Traces

Problem

Given $f(f(n)) = 2n$ and $f(4n+1) = 4n+3$, find $f(1000)$.
Ground truth: $\boxed{1016}$.

Unmasked Expert Hint

The unmasked expert hint exposes the decisive numeric path and the final answer-relevant computation:

```
Decompose 1000 = 2^3 * 125.
125 is odd and of the form 4k + 1
(125 = 4 * 31 + 1), so f(125) = 4 * 31 + 3 = 127.
Therefore, f(1000) = 2^3 * f(125) = 8 * 127 = 1016.
```

Thus the hint already contains the target decomposition, the intermediate value $f(125) = 127$, and the final answer 1016.

Expert Output

After conditioning on the unmasked expert trace, the rollout does not show an independent derivation. It simply emits the final boxed answer:

$$\boxed{1016}$$

GRPO Output

Without expert guidance, the rollout does not recover the useful decomposition rule. It repeatedly tries to solve the problem through a preimage search:

```
Let's denote f(1000) = k. Then, using the first property, we have f(k) = 2000.

We need to find k such that f(k) = 2000. Since 2000 is even, we need to find an odd
    number m such that f(m) = 2001. Using the second property, we have f(4k + 1) = 4k +
    3. So, we need to find k such that 4k + 3 = 2001.

Solving for k:
4k + 3 = 2001
4k = 1998
k = 499.5

Since k must be an integer, we need to adjust our approach.
```

The rollout repeats this failed pattern, repeatedly forcing even targets such as 2000 into a rule that applies to inputs of the form $4n + 1$.

SMEPO Output

The masked trace removes the concrete numbers but preserves the decomposition template. The SMEPO rollout reconstructs the missing numeric content and derives the scaling rule it uses:

```
1000 is an even number, and we can express it as
1000 = 2^3 * 125. Here, 125 is odd and can be written as
```

```
125 = 4 * 31 + 1.

Since 125 = 4 * 31 + 1, we have
f(125) = 4 * 31 + 3 = 127.

First, derive the scaling property. From f(f(n)) = 2n,
applying the same identity to f(n) gives
f(f(f(n))) = 2f(n). Since f(f(n)) = 2n, the left-hand side
is also f(2n). Hence f(2n) = 2f(n), and by induction
f(2^k * m) = 2^k * f(m).

Since 1000 = 2^3 * 125, we get
f(1000) = f(2^3 * 125) = 2^3 * f(125)
        = 8 * 127 = 1016.
```

$\boxed{1016}$

Analysis

This case shows why the difference between the hint and the rollout matters. The unmasked expert hint contains the answer-relevant computation, including $1000 = 2^3 \cdot 125$, $125 = 4 \cdot 31 + 1$, $f(125) = 127$, and $8 \cdot 127 = 1016$. Once this information is visible, the resulting Expert rollout can obtain reward by directly emitting $\boxed{1016}$. The rollout is correct, but it does not demonstrate that the model has learned or derived a reusable reasoning procedure.

The vanilla GRPO rollout fails for a specific reason: it tries to invert f by setting $f(1000) = k$ and then matching $f(k) = 2000$ to the rule $f(4n + 1) = 4n + 3$. This repeatedly produces nonintegral or mismatched congruence conditions, such as $4k + 3 = 2001$. The missing insight is not just the final numeric answer, but the structural reduction: first derive $f(2n) = 2f(n)$ from $f(f(n)) = 2n$, then use induction to obtain $f(2^k m) = 2^k f(m)$, and finally reduce 1000 to its odd component 125.

SMEPO keeps this useful structural plan while masking the concrete answer-relevant values. The rollout still has to recover $1000 = 2^3 \cdot 125$, identify $125 = 4 \cdot 31 + 1$, compute $f(125) = 127$, derive the scaling rule $f(2^k m) = 2^k f(m)$, and combine these steps to obtain 1016. Thus the masked trace provides the reasoning route without directly exposing the reward-relevant numeric path as an answer channel.

Annotated Summary: Code Synthesis with Expert Traces

Problem

Given N points in a plane, choose a nonzero direction (p, q). Points can be collected at zero additional cost when the previously collected point is exactly one step (p, q) behind. The goal is to choose (p, q) to minimize the number of chains needed to collect all points.

Unmasked Expert Hint

The unmasked expert hint exposes a complete executable solution. Its core idea is to enumerate all offsets induced by point pairs and count how many chains start under each offset:

```
points_set = set(points)
diff_set = set()

for i in range(n):
    x_i, y_i = points[i]
    for j in range(n):
        if i != j:
            x_j, y_j = points[j]
            dx = x_j - x_i
            dy = y_j - y_i
            diff_set.add((dx, dy))

min_chains = None
```

```
for dx, dy in diff_set:
    current = 0
    for (x, y) in points:
        if (x - dx, y - dy) not in points_set:
            current += 1
    if min_chains is None or current < min_chains:
        min_chains = current

print(min_chains)
```

Expert Output

After conditioning on the unmasked expert trace, the rollout directly emits the solution implementation. The output is essentially the expert algorithm:

```
n = int(input())
points = [tuple(map(int, input().split())) for _ in range(n)]

if n == 1:
    print(1)
    exit()

points_set = set(points)
diff_set = set()

for i in range(n):
    x_i, y_i = points[i]
    for j in range(n):
        if i != j:
            x_j, y_j = points[j]
            dx = x_j - x_i
            dy = y_j - y_i
            diff_set.add((dx, dy))

min_chains = None

for dx, dy in diff_set:
    current = 0
    for (x, y) in points:
        if (x - dx, y - dy) not in points_set:
            current += 1
    if min_chains is None or current < min_chains:
        min_chains = current

print(min_chains)
```

GRPO Output

The vanilla GRPO rollout instead produces an invalid bounded search over small (p, q) values:

```
min_cost = float('inf')
for p in range(-10, 11):
    for q in range(-10, 11):
        if p == 0 and q == 0:
            continue
        cost = 0
        collected = set()
        for i in range(N):
            for j in range(i + 1, N):
                if (balls[i][0], balls[i][1]) in collected and \
                   (balls[j][0], balls[j][1]) in collected:
                    continue
                if (balls[i][0] - balls[j][0]) != \
                   p * (balls[i][1] - balls[j][1]) // q or \
                   (balls[i][1] - balls[j][1]) % q != 0:
                    cost += 1
```

```
                    collected.add((balls[i][0], balls[i][1]))
                    collected.add((balls[j][0], balls[j][1]))
            min_cost = min(min_cost, cost)
print(min_cost)
```
This code is not equivalent to the chain-cover objective.

SMEPO Output

The masked trace hides the full code block but keeps the algorithmic procedure. The SMEPO rollout reconstructs the key logic: enumerate point-pair offsets, then count points without predecessors.

```
points_set = set(tuple(point) for point in points)

if n == 1:
    print(1)
    return

min_cost = float('inf')
dx_dy_pairs = set()

for i in range(n):
    for j in range(i + 1, n):
        dx = points[j][0] - points[i][0]
        dy = points[j][1] - points[i][1]
        dx_dy_pairs.add((dx, dy))

for dx, dy in dx_dy_pairs:
    chains = 0
    for x, y in points:
        if (x - dx, y - dy) not in points_set:
            chains += 1
    min_cost = min(min_cost, chains)

print(min_cost)
```

Analysis

This case highlights a code-specific form of reward-relevant content. In code synthesis, the unmasked expert hint does not merely reveal an intermediate clue; it contains a complete executable implementation. The resulting Expert output directly emits the same algorithm. This is correct, but the reward can be achieved by reproducing the exposed code body rather than by learning the underlying programming strategy.

The vanilla GRPO output illustrates a different failure mode. It treats the direction (p, q) as something that can be brute-forced in the bounded range $[-10, 10]^2$, although the coordinates can be as large as 10^9 and the optimal direction can be any offset between two input points. It also uses a slope-like integer-division test involving q, which is not the same as checking whether one point is exactly one step (p, q) after another. This can fail for horizontal directions when $q = 0$ and is conceptually wrong even when $q \neq 0$. Finally, it increments cost inside a pairwise loop with a "collected" set, whereas the correct objective for a fixed offset is to count chain starts: points whose predecessor $(x - p, y - q)$ is absent.

SMEPO masks the directly copyable code body while preserving the procedural outline. The generated solution is not necessarily textually identical to the expert, but it recovers the essential invariant: for each candidate offset, every chain contributes exactly one start point. This is the intended effect of masking in code: remove the executable answer while retaining enough guidance for the model to synthesize the algorithm.

Annotated Summary: Agentic Search with Expert Traces

Problem

What age was the American social entrepreneur who was one of eight killed in the 2015 Amtrak train derailment from Washington, D.C. to New York City?
Ground truth: $\boxed{39}$.

Unmasked Expert Hint

The unmasked expert hint reveals both the target entity and the answer:

```
Round 2:
From the list of victims, Rachel Jacobs was identified as the CEO of a technology
    education company, which aligns with the description of a social entrepreneur.
    Next, I'll search for her age at the time of the incident.

<search> Find the age of Rachel Jacobs at the time of the 2015 Amtrak train derailment.
    </search>

Round 3:
Based on the information, Rachel Jacobs was 39 years old at the time of the 2015 Amtrak
    train derailment. She was the American social entrepreneur among the victims.

<answer>39</answer>
```

Expert Output

After conditioning on the unmasked expert trace, the rollout directly emits the answer:

```
Answer:  <answer>39</answer>
```

GRPO Output

The vanilla GRPO rollout identifies the derailment but repeatedly searches for generic victim lists. It fails to pivot to the discriminative entity description:

```
<search> Amtrak derailment 2015 </search>

From Doc 1, I see that the Amtrak derailment in 2015 involved a train traveling from
    Washington, D.C. to New York City and resulted in 8 deaths and over 200 injuries.
    However, I don't see the names of the individuals who were killed.

<search> Amtrak derailment 2015 victims </search>

From Doc 1, I see that 8 people were killed in the derailment, but I still don't have
    the names of the individuals. I will need to search for a more detailed list of the
    victims.

<search> names of the 8 people killed in the 2015 Amtrak derailment </search>

From Doc 1, I still don't have the names of the individuals. I will need to search for a
    more detailed list of the victims.

<search> 2015 Amtrak derailment victim list </search>

From Doc 1, I still don't have the names of the individuals. I will need to search for a
    more detailed list of the victims.

<search> 2015 Philadelphia train derailment victim names </search>
```

SMEPO Output

The masked trace removes answer-relevant entities such as the target name and age, but preserves the search plan. The SMEPO rollout reformulates the second query around the entity description:

```
<search> Find information about the 2015 Amtrak train derailment from Washington, D.C.
    to New York City, including the list of victims killed in the incident. </search>

From the information, I've found that the Amtrak train derailment occurred on May 12,
    2015, and 8 people were killed, including a social entrepreneur. Next, I'll search
    for the name of the social entrepreneur and her age at the time of the incident.

<search> Find the name of the American social entrepreneur who was one of the eight
    killed in the 2015 Amtrak train derailment from Washington, D.C. to New York City,
    and her age at the time of the incident. </search>
```

The retrieved evidence identifies Rachel Jacobs and states that she was killed at age 39. The rollout returns:

<answer>39</answer>.

Analysis

This case cleanly separates answer exposure from useful search procedure. The unmasked expert hint directly reveals the target entity, Rachel Jacobs, and the final age, 39. The resulting Expert rollout is therefore extremely short: it simply emits <answer>39</answer>. This is precisely the kind of reward-relevant channel that can make full expert traces problematic. The policy can achieve reward by finalizing exposed information rather than by performing the search.

The vanilla GRPO rollout finds the correct event but fails to use the key constraint in the question. It repeatedly searches for generic victim lists, such as "Amtrak derailment 2015 victims" and "names of the 8 people killed." These searches return descriptions of the derailment but do not surface the needed entity. The rollout loops over near-duplicate search intents instead of pivoting to the more specific clue: the victim was an American social entrepreneur.

SMEPO removes the target entity and final answer from the trace, but preserves the useful action sequence. After identifying the derailment, the model learns to reformulate the query around the discriminative entity description and the desired attribute: the name of the American social entrepreneur and her age. This query retrieves the target entity and supports the final answer 39. Thus the masked trace guides the search strategy without directly revealing the reward-relevant entity.

D.2 Full Case Study Rollouts

We then include the full rollout versions of the same case studies for completeness and transparency. The annotated summaries above quote only the most relevant spans, while the full rollouts preserve the surrounding context, including the original hints and model outputs.

Full Rollout: Functional Equation Reasoning

Question

```
The function $f(n)$ is defined on the positive integers such that $f(f(n)) = 2n$ and
    $f(4n + 1) = 4n + 3$ for all positive integers $n.$  Find $f(1000).$
```

Ground Truth

```
1016
```

Unmasked Expert Hint

```
To solve for \( f(1000) \), we analyze the function \( f \) defined by the conditions \(
    f(f(n)) = 2n \) and \( f(4n + 1) = 4n + 3 \).

1. **Odd Numbers**:
   - For \( n = 4k + 1 \), \( f(n) = 4k + 3 \).
   - For \( n = 4k + 3 \), \( f(n) = 8k + 2 \).
```

```
2. **Even Numbers**:
   - If \( n \) is even, express \( n \) as \( 2^k \cdot m \) where \( m \) is odd. Then
     \( f(n) = 2^k \cdot f(m) \).

**Application to \( n = 1000 \)**:
- Decompose \( 1000 = 2^3 \cdot 125 \).
- \( 125 \) is odd and of the form \( 4k + 1 \) (\( 125 = 4 \cdot 31 + 1 \)), so \(
    f(125) = 4 \cdot 31 + 3 = 127 \).
- Therefore, \( f(1000) = 2^3 \cdot f(125) = 8 \cdot 127 = 1016 \).

Verification:
- \( f(1016) = 2^3 \cdot f(127) \). Since \( 127 = 4 \cdot 31 + 3 \), \( f(127) = 8
    \cdot 31 + 2 = 250 \).
- Thus, \( f(1016) = 8 \cdot 250 = 2000 \), confirming \( f(f(1000)) = 2 \cdot 1000 \).
```

Masked Expert Hint

```
To solve for \( f([NUMBER]) \), we analyze the function \( f \) defined by the
    conditions \( f(f(n)) = [NUMBER]n \) and \( f([NUMBER]n + [NUMBER]) = [NUMBER]n +
    [NUMBER] \).

**Odd Numbers**:
   - For \( n = [NUMBER]k + [NUMBER] \), \( f(n) = [NUMBER]k + [NUMBER] \).
   - For \( n = [NUMBER]k + [NUMBER] \), \( f(n) = [NUMBER]k + [NUMBER] \).

**Even Numbers**:
   - If \( n \) is even, express \( n \) as \( [NUMBER]^k \cdot m \) where \( m \) is
     odd. Then \( f(n) = [NUMBER]^k \cdot f(m) \).

**Application to \( n = [NUMBER] \)**:
- Decompose \( [NUMBER] = [NUMBER]^[NUMBER] \cdot [NUMBER] \).
- \( [NUMBER] \) is odd and of the form \( [NUMBER]k + [NUMBER] \) (\( [NUMBER] =
    [NUMBER] \cdot [NUMBER] + [NUMBER] \)), so \( f([NUMBER]) = [NUMBER] \cdot [NUMBER]
    + [NUMBER] = [NUMBER] \).
- Therefore, \( f([NUMBER]) = [NUMBER]^[NUMBER] \cdot f([NUMBER]) = [NUMBER] \cdot
    [NUMBER] = [NUMBER] \).

Verification:
- \( f([NUMBER]) = [NUMBER]^[NUMBER] \cdot f([NUMBER]) \). Since \( [NUMBER] = [NUMBER]
    \cdot [NUMBER] + [NUMBER] \), \( f([NUMBER]) = [NUMBER] \cdot [NUMBER] + [NUMBER] =
    [NUMBER] \).
- Thus, \( f([NUMBER]) = [NUMBER] \cdot [NUMBER] = [NUMBER] \), confirming \(
    f(f([NUMBER])) = [NUMBER] \cdot [NUMBER] \).
```

Expert Output

```
\boxed{1016}
```

GRPO Output

```
To solve the problem, we need to determine the value of \( f(1000) \) given the
    properties of the function \( f \):
1. \( f(f(n)) = 2n \)
2. \( f(4n + 1) = 4n + 3 \)

Let's proceed step by step.

### Step 1: Understand the function properties
- The first property \( f(f(n)) = 2n \) tells us that applying the function twice
    doubles the input.
- The second property \( f(4n + 1) = 4n + 3 \) gives us the value of the function for
    numbers of the form \( 4n + 1 \).
```

```
### Step 2: Find a pattern or a way to express \( f(n) \) for other values
We need to find \( f(1000) \). Notice that 1000 is not of the form \( 4n + 1 \), so we
     need to use the first property to find a way to express \( f(1000) \).

Let's denote \( f(1000) = k \). Then, using the first property, we have \( f(k) = 2000
     \).

Now, we need to find \( k \). Since \( k \) is not of the form \( 4n + 1 \), we need to
     find a number \( m \) such that \( f(m) = 1000 \). Then, using the first property,
     we have \( f(f(m)) = 2m \), so \( f(1000) = 2m \).

To find \( m \), we need to express 1000 in a form that we can use the second property.
     Notice that 1000 is even, so it is not directly useful with the second property.
     However, we can use the first property to find a relationship.

Since \( f(f(n)) = 2n \), we can try to find a number \( n \) such that \( f(n) = 1000
     \). Let's denote \( f(n) = 1000 \). Then, using the first property, we have \(
     f(1000) = 2n \).

We need to find \( n \) such that \( f(n) = 1000 \). Since 1000 is even, we need to find
     an odd number \( m \) such that \( f(m) = 1001 \) (the closest odd number to 1000).
     Using the second property, we have \( f(4k + 1) = 4k + 3 \). So, we need to find \(
     k \) such that \( 4k + 3 = 1001 \).

Solving for \( k \):
\[ 4k + 3 = 1001 \]
\[ 4k = 998 \]
\[ k = 249.5 \]

Since \( k \) must be an integer, we need to adjust our approach. We can use the fact
     that \( f(f(n)) = 2n \) to find a direct relationship. Let's denote \( f(1000) = k
     \). Then, \( f(k) = 2000 \).

We need to find \( k \) such that \( f(k) = 2000 \). Since 2000 is even, we need to find
     an odd number \( m \) such that \( f(m) = 2001 \). Using the second property, we
     have \( f(4k + 1) = 4k + 3 \). So, we need to find \( k \) such that \( 4k + 3 =
     2001 \).

Solving for \( k \):
\[ 4k + 3 = 2001 \]
\[ 4k = 1998 \]
\[ k = 499.5 \]

Since \( k \) must be an integer, we need to adjust our approach. We can use the fact
     that \( f(f(n)) = 2n \) to find a direct relationship. Let's denote \( f(1000) = k
     \). Then, \( f(k) = 2000 \).

We need to find \( k \) such that \( f(k) = 2000 \). Since 2000 is even, we need to find
     an odd number \( m \) such that \( f(m) = 2001 \). Using the second property, we
     have \( f(4k + 1) = 4k + 3 \). So, we need to find \( k \) such that \( 4k + 3 =
     2001 \).

Solving for \( k \):
\[ 4k + 3 = 2001 \]
\[ 4k = 1998 \]
\[ k = 499.5 \]

...

Since \( k \) must be an integer, we need to adjust our approach. We can use the fact
     that \( f(f(n)) = 2n \) to find a direct relationship. Let's denote \( f(1000) = k
     \). Then, \( f(k) = 2000 \).

We need to find \( k \) such that \( f(k) = 2000 \). Since 2000 is even, we need to find
     an odd number \( m \) such that \( f(m) = 2001 \). Using the second property, we
```

```
    have \( f(4k + 1) = 4k + 3 \). So, we need to find \( k \) such that \( 4k + 3 =
    2001 \).

Solving for \( k \):
\[ 4k + 3 = 2001 \]
\[ 4k = 1998 \]
\[ k = 499.5 \]

Since \( k \) must be an integer, we need to adjust our approach. We can use the fact
    that \( f(f(n)) = 2n \) to find a direct relationship. Let's denote \( f(1000) = k
    \). Then, \( f(k) = 2000 \).
```

SMEPO Output

```
To find \( f(1000) \), we need to understand the behavior of the function \( f \) given
    the conditions \( f(f(n)) = 2n \) and \( f(4n + 1) = 4n + 3 \).

1. **Identify the form of \( n \):**
   - \( 1000 \) is an even number, and we can express it as \( 1000 = 2^3 \cdot 125 \).
     Here, \( 125 \) is odd and can be written as \( 125 = 4 \cdot 31 + 1 \).

2. **Use the given conditions:**
   - Since \( 125 = 4 \cdot 31 + 1 \), we have \( f(125) = 4 \cdot 31 + 3 = 127 \).

3. **Determine \( f(1000) \):**
   - First, derive the scaling property. From \( f(f(n)) = 2n \), applying the same
     identity to \( f(n) \) gives \( f(f(f(n))) = 2f(n) \). Since \( f(f(n)) = 2n \),
     the left-hand side is also \( f(2n) \). Hence \( f(2n) = 2f(n) \), and by induction
     \( f(2^k \cdot m) = 2^k \cdot f(m) \).
   - Since \( 1000 = 2^3 \cdot 125 \), we get \( f(1000) = f(2^3 \cdot 125) = 2^3 \cdot
     f(125) = 8 \cdot 127 = 1016 \).

Therefore, the final answer is \boxed{1016}.
```

Outcome

Expert: Correct **GRPO:** Wrong **SMEPO:** Correct

Full Rollout: Chain Counting Over Candidate Directions

Question

```
Solve the following programming problem.
Your program should read from standard input and write to standard output.

There are N balls in a two-dimensional plane. The i-th ball is at coordinates (x_i, y_i).

We will collect all of these balls, by choosing two integers p and q such that p \neq 0
    or q \neq 0 and then repeating the following operation:

* Choose a ball remaining in the plane and collect it. Let (a, b) be the coordinates of
    this ball. If we collected a ball at coordinates (a - p, b - q) in the previous
    operation, the cost of this operation is 0. Otherwise, including when this is the
    first time to do this operation, the cost of this operation is 1.



Find the minimum total cost required to collect all the balls when we optimally choose p
    and q.

Constraints
```

```
* 1 \leq N \leq 50
* |x_i|, |y_i| \leq 10^9
* If i \neq j, x_i \neq x_j or y_i \neq y_j.
* All values in input are integers.

Input

Input is given from Standard Input in the following format:


N
x_1 y_1
:
x_N y_N


Output

Print the minimum total cost required to collect all the balls.

Examples

Input

2
1 1
2 2


Output

1


Input

3
1 4
4 6
7 8


Output

1


Input

4
1 1
1 2
2 1
2 2


Output

2
```

Ground Truth


```
{"inputs": ["3\n1 4\n4 6\n7 8", "4\n1 1\n1 2\n2 1\n2 2", "2\n1 1\n2 2", "3\n1 4\n4 6\n11
    8", "2\n1 2\n2 2", "4\n1 1\n1 4\n2 1\n2 2", "3\n1 4\n4 6\n11 3", "3\n0 4\n4 6\n11
    3", "3\n-1 4\n4 6\n11 3", "3\n-2 4\n4 6\n11 3", "3\n-2 4\n5 6\n11 3", "3\n-2 4\n5
```

```
6\n11 4", "3\n-1 4\n5 6\n11 4", "3\n-1 4\n5 10\n11 4", "3\n-1 7\n5 10\n11 4",
"3\n-1 7\n5 12\n11 4", "3\n-1 7\n5 12\n10 4", "3\n-2 7\n5 12\n10 4", "3\n-2 7\n5
16\n10 4", "3\n-2 0\n5 16\n10 4", "3\n-2 -1\n5 16\n10 4", "3\n0 0\n5 16\n10 4",
"3\n-1 0\n5 16\n10 4", "3\n-1 0\n5 32\n10 4", "3\n-1 1\n5 32\n10 4", "3\n-1 2\n5
32\n10 4", "3\n-1 4\n5 32\n10 4", "3\n-1 4\n5 32\n10 2", "3\n-1 4\n5 32\n10 0",
"3\n-1 4\n5 32\n20 0", "3\n-1 4\n9 32\n20 0", "3\n-1 4\n9 32\n24 0", "3\n-1 4\n9
34\n24 0", "3\n-1 6\n9 34\n24 0", "3\n-1 7\n9 34\n24 0", "3\n-2 7\n9 34\n24 0",
"3\n-4 7\n9 34\n24 0", "3\n-4 7\n9 34\n24 -1", "3\n0 7\n9 34\n24 -1", "3\n0 7\n9
33\n24 -1", "3\n0 7\n9 33\n24 0", "3\n0 7\n10 33\n24 0", "3\n0 7\n10 33\n24 -1",
"3\n0 7\n10 33\n1 -1", "3\n0 4\n10 33\n1 -1", "3\n0 4\n10 33\n2 -1", "3\n0 4\n10
33\n3 -1", "3\n0 4\n2 33\n3 -1", "3\n0 4\n1 33\n3 -1", "3\n0 4\n1 17\n3 -1", "3\n0
1\n1 17\n3 -1", "3\n0 1\n1 17\n6 -1", "3\n0 1\n1 17\n6 0", "3\n0 1\n0 17\n6 0",
"3\n1 1\n0 17\n6 0", "3\n1 1\n0 17\n2 0", "3\n0 1\n0 17\n2 0", "3\n0 1\n0 25\n2 0",
"3\n0 1\n0 25\n4 0", "3\n0 1\n0 25\n6 0", "3\n0 1\n0 32\n6 0", "3\n0 1\n1 32\n6 0",
"3\n1 1\n1 32\n6 0", "3\n1 1\n1 32\n6 1", "3\n1 0\n1 32\n6 1", "3\n1 0\n1 32\n6 2",
"3\n1 0\n0 32\n6 2", "3\n1 0\n0 32\n6 1", "3\n2 0\n0 32\n6 1", "3\n2 1\n0 32\n6 1",
"3\n1 4\n3 6\n7 8", "2\n2 1\n2 2", "3\n1 4\n4 6\n8 8", "2\n1 2\n2 4", "3\n2 4\n4
6\n11 3", "3\n0 4\n0 6\n11 3", "3\n-1 4\n4 6\n11 2", "3\n-2 5\n4 6\n11 3", "3\n-2
4\n5 3\n11 3", "3\n-2 4\n5 10\n11 4", "3\n-1 4\n5 6\n1 4", "3\n-2 4\n5 2\n11 4",
"3\n-1 7\n5 10\n21 4", "3\n-1 7\n5 12\n11 2", "3\n-1 7\n5 3\n10 4", "3\n-2 7\n6
12\n10 4", "3\n-2 7\n4 16\n10 4", "3\n-4 0\n5 16\n10 4", "3\n-2 -1\n10 16\n10 4",
"3\n1 0\n5 16\n10 4", "3\n-1 0\n5 2\n10 4", "3\n-1 0\n5 32\n20 4", "3\n-1 0\n6
32\n10 4", "3\n-1 2\n5 33\n10 4", "3\n-1 4\n5 32\n10 3", "3\n-1 4\n5 32\n9 2",
"3\n-1 3\n5 32\n10 0", "3\n-1 4\n5 49\n20 0", "3\n-1 4\n9 44\n20 0", "3\n-1 4\n9
19\n24 0", "3\n-1 4\n9 34\n45 0", "3\n-1 6\n9 34\n10 0", "3\n-1 7\n9 34\n24 -1"],
"outputs": ["1", "2", "1", "2\n", "1\n", "3\n", "2\n", "2\n", "2\n", "2\n", "2\n",
"2\n", "2\n", "2\n", "2\n", "2\n", "2\n", "2\n", "2\n", "2\n", "2\n", "2\n", "2\n",
"2\n", "2\n", "2\n", "2\n", "2\n", "2\n", "2\n", "2\n", "2\n", "2\n", "2\n", "2\n",
"2\n", "2\n", "2\n", "2\n", "2\n", "2\n", "2\n", "2\n", "2\n", "2\n", "2\n", "2\n",
"2\n", "2\n", "2\n", "2\n", "2\n", "2\n", "2\n", "2\n", "2\n", "2\n", "2\n", "2\n",
"2\n", "2\n", "2\n", "2\n", "2\n", "2\n", "2\n", "2\n", "2\n", "2\n", "2\n", "2\n",
"1\n", "2\n", "1\n", "2\n", "2\n", "2\n", "2\n", "2\n", "2\n", "2\n", "2\n", "2\n",
"2\n", "2\n", "2\n", "2\n", "2\n", "2\n", "2\n", "2\n", "2\n", "2\n", "2\n", "2\n",
"2\n", "2\n", "2\n", "2\n", "2\n", "2\n", "2\n", "2\n"]}
```


Unmasked Expert Hint

To solve this problem, we need to determine the minimum total cost required to collect all balls in a 2D plane by optimally choosing two integers p and q (not both zero). The cost is minimized by arranging the balls into the fewest possible chains, where each chain consists of consecutive points differing by (p, q).

Approach
1. **Input Handling**: Read the number of balls and their coordinates.
2. **Special Case for Single Ball**: If there is only one ball, the cost is always 1.
3. **Generate Differences**: Compute all possible (dx, dy) pairs from the differences between every pair of distinct points.
4. **Determine Chains**: For each (dx, dy) pair, calculate the number of chains required. A chain starts at a point if the previous point (by subtracting dx and dy) is not present.
5. **Find Minimum Cost**: Iterate through all (dx, dy) pairs to find the minimum number of chains, which corresponds to the minimum cost.

Solution Code
```python
n = int(input())
points = [tuple(map(int, input().split())) for _ in range(n)]

if n == 1:
    print(1)
    exit()

points_set = set(points)
diff_set = set()
```

```
for i in range(n):
    x_i, y_i = points[i]
    for j in range(n):
        if i != j:
            x_j, y_j = points[j]
            dx = x_j - x_i
            dy = y_j - y_i
            diff_set.add((dx, dy))

min_chains = None

for dx, dy in diff_set:
    current = 0
    for (x, y) in points:
        if (x - dx, y - dy) not in points_set:
            current += 1
    if min_chains is None or current < min_chains:
        min_chains = current

print(min_chains)
```

Explanation
1. **Input Handling**: The code reads the number of balls and their coordinates, storing them in a list and a set for quick lookups.
2. **Special Case Handling**: If there's only one ball, the cost is directly output as 1.
3. **Generate Differences**: For each pair of distinct points, the differences (dx, dy) are computed and stored in a set to avoid duplicates.
4. **Chain Calculation**: For each (dx, dy) pair, the code checks each point to see if it can start a new chain by verifying if the previous point (x-dx, y-dy) exists in the set of points. The count of such starting points gives the number of chains.
5. **Find Minimum**: The minimum number of chains across all (dx, dy) pairs is determined and printed as the result.

Masked Expert Hint

To solve this problem, we need to determine the minimum total cost required to collect all balls in a 2D plane by optimally choosing two integers p and q (not both zero). The cost is minimized by arranging the balls into the fewest possible chains, where each chain consists of consecutive points differing by (p, q).

Approach
1. **Input Handling**: Read the number of balls and their coordinates.
2. **Special Case for Single Ball**: If there is only one ball, the cost is always 1.
3. **Generate Differences**: Compute all possible (dx, dy) pairs from the differences between every pair of distinct points.
4. **Determine Chains**: For each (dx, dy) pair, calculate the number of chains required. A chain starts at a point if the previous point (by subtracting dx and dy) is not present.
5. **Find Minimum Cost**: Iterate through all (dx, dy) pairs to find the minimum number of chains, which corresponds to the minimum cost.

Solution Code
```python
[CODE]
```

Explanation
1. **Input Handling**: The code reads the number of balls and their coordinates, storing them in a list and a set for quick lookups.
2. **Special Case Handling**: If there's only one ball, the cost is directly output as 1.
3. **Generate Differences**: For each pair of distinct points, the differences (dx, dy) are computed and stored in a set to avoid duplicates.

```
4. **Chain Calculation**: For each (dx, dy) pair, the code checks each point to see if
     it can start a new chain by verifying if the previous point (x-dx, y-dy) exists in
     the set of points. The count of such starting points gives the number of chains.
5. **Find Minimum**: The minimum number of chains across all (dx, dy) pairs is
     determined and printed as the result.
```

Expert Output

````
```python
n = int(input())
points = [tuple(map(int, input().split())) for _ in range(n)]

if n == 1:
    print(1)
    exit()

points_set = set(points)
diff_set = set()

for i in range(n):
    x_i, y_i = points[i]
    for j in range(n):
        if i != j:
            x_j, y_j = points[j]
            dx = x_j - x_i
            dy = y_j - y_i
            diff_set.add((dx, dy))

min_chains = None

for dx, dy in diff_set:
    current = 0
    for (x, y) in points:
        if (x - dx, y - dy) not in points_set:
            current += 1
    if min_chains is None or current < min_chains:
        min_chains = current

print(min_chains)
```
````

GRPO Output

````
```` python
import sys

def main():
    N = int(input())
    balls = []
    for _ in range(N):
        balls.append(list(map(int, input().split())))

    min_cost = float('inf')
    for p in range(-10, 11):
        for q in range(-10, 11):
            if p == 0 and q == 0:
                continue
            cost = 0
            collected = set()
            for i in range(N):
                for j in range(i + 1, N):
                    if (balls[i][0], balls[i][1]) in collected and (balls[j][0],
     balls[j][1]) in collected:
                        continue
````
````

```
                    if (balls[i][0] - balls[j][0]) != p * (balls[i][1] - balls[j][1]) //
    q or (balls[i][1] - balls[j][1]) % q != 0:
                        cost += 1
                        collected.add((balls[i][0], balls[i][1]))
                        collected.add((balls[j][0], balls[j][1]))
            min_cost = min(min_cost, cost)
    print(min_cost)

if __name__ == "__main__":
    main()
‘‘\‘
```

SMEPO Output

```
‘‘\‘python
import sys
import numpy as np

def main():
    n = int(sys.stdin.readline())
    points = [list(map(int, sys.stdin.readline().split())) for _ in range(n)]
    points_set = set(tuple(point) for point in points)

    if n == 1:
        print(1)
        return

    min_cost = float('inf')
    dx_dy_pairs = set()

    for i in range(n):
        for j in range(i+1, n):
            dx, dy = points[j][0] - points[i][0], points[j][1] - points[i][1]
            dx_dy_pairs.add((dx, dy))

    for dx, dy in dx_dy_pairs:
        chains = 0
        for x, y in points:
            if (x - dx, y - dy) not in points_set:
                chains += 1
        min_cost = min(min_cost, chains)

    print(min_cost)

if __name__ == "__main__":
    main()
‘‘\‘
```

Outcome

Expert: Correct **GRPO:** Wrong **SMEPO:** Correct

Full Rollout: Entity-Specific Web Search

Question

```
What age was the American social entrepeneur who was one of eight who got killed in 2015
     when an Amtrak train travelling from Washington, D.C. to New York City derailed?
```

Ground Truth

```
39
```

Unmasked Expert Hint

```
Round 1: Let's break this down step by step:

1. Identify the Amtrak train derailment incident in 2015 involving a train from
     Washington, D.C. to New York City
2. Find the list of eight victims killed in that incident
3. Determine which of those victims was an American social entrepreneur
4. Find the age of that individual at the time of the incident

First, I'll search for information about the 2015 Amtrak derailment to identify the
     victims.

<search> Find information about the 2015 Amtrak train derailment from Washington, D.C.
     to New York City, including the list of victims killed in the incident. </search>
Round 2: From the list of victims, Rachel Jacobs was identified as the CEO of a
     technology education company, which aligns with the description of a social
     entrepreneur. Next, I'll search for her age at the time of the incident.

<search> Find the age of Rachel Jacobs at the time of the 2015 Amtrak train derailment.
     </search>
Round 3: Based on the information, Rachel Jacobs was 39 years old at the time of the
     2015 Amtrak train derailment. She was the American social entrepreneur among the
     victims.

<answer>39</answer>
```

Masked Expert Hint

```
Round 1: Let's break this down step by step:

1. Identify the [ENTITY] train derailment incident in [ENTITY] involving a train from
     [ENTITY], [ENTITY] to [ENTITY]
2. Find the list of [ENTITY] victims killed in that incident
3. Determine which of those victims was an [ENTITY] social entrepreneur
4. Find the age of that individual at the time of the incident

First, I'll search for information about the [ENTITY] [ENTITY] derailment to identify
     the victims.

<search> Find information about the [ENTITY] [ENTITY] train derailment from [ENTITY],
     [ENTITY] to [ENTITY], including the list of victims killed in the incident.
     </search>
Round 2: From the list of victims, [ENTITY] was identified as the CEO of a technology
     education company, which aligns with the description of a social entrepreneur.
     Next, I'll search for her age at the time of the incident.

<search> Find the age of [ENTITY] at the time of the [ENTITY] [ENTITY] train derailment.
     </search>
Round 3: Based on the information, [ENTITY] was [ENTITY] years old at the time of the
     [ENTITY] [ENTITY] train derailment. She was the [ENTITY] social entrepreneur among
     the victims.

<answer>[ENTITY]</answer>
```

Expert Output

```
Answer: <answer>39</answer>
```

GRPO Output

```
To answer this question, I need to identify the American social entrepreneur who was
     killed in the Amtrak derailment in 2015 and then determine their age at the time of
     the accident.

First, I will search for information about the Amtrak derailment in 2015 to find out who
     was involved.

<search> Amtrak derailment 2015 </search>

<information>Doc 1(Title: "2015 Philadelphia train derailment") 2015 Philadelphia train
     derailment On May 12, 2015, an Amtrak ""Northeast Regional"" train from Washington,
     D.C. bound for New York City derailed and wrecked on the Northeast Corridor in the
     Port Richmond neighborhood of Philadelphia, Pennsylvania. Of 238 passengers and 5
     crew on board, 8 were killed and over 200 injured, 11 critically. The train was
     traveling at in a zone of curved tracks when it derailed. Some of the passengers
     had to be extricated from the wrecked cars. Many of the passengers and local
     residents helped first responders during the rescue operation. Five local hospitals
     treated the injured. The
Doc 2(Title: "2015 Philadelphia train derailment") 6, 2018, following an appeal of
     Gehret's ruling by the Pennsylvania Attorney General's Office, Philadelphia Common
     Pleas Court Judge Kathryn Streeter Lewis reinstated all criminal charges against
     Bostian. 2015 Philadelphia train derailment On May 12, 2015, an Amtrak ""Northeast
     Regional"" train from Washington, D.C. bound for New York City derailed and wrecked
     on the Northeast Corridor in the Port Richmond neighborhood of Philadelphia,
     Pennsylvania. Of 238 passengers and 5 crew on board, 8 were killed and over 200
     injured, 11 critically. The train was traveling at in a zone of curved tracks when
     it derailed. Some of the passengers had
Doc 3(Title: "2015 Philadelphia train derailment") the curve at which the derailment
     occurred, and that also would have limited the train's speed entering the curve.
     Shortly after the derailment, Amtrak completed ATC installation on the northbound
     track. The 2015 wreck was the deadliest on the Northeast Corridor since 1987, when
     16 people died in a wreck near Baltimore. At about 9:10 p.m. (EDT) on Tuesday, May
     12, 2015, Amtrak's northbound ""Northeast Regional"" 188 departed Philadelphia's
     30th Street Station en route from Washington, D.C., to New York City. The train
     consisted of seven cars hauled by a year-old Amtrak Cities Sprinter (ACS)-64
     locomotive, No. 601. The engineer</information>

From Doc 1, I see that the Amtrak derailment in 2015 involved a train traveling from
     Washington, D.C. to New York City and resulted in 8 deaths and over 200 injuries.
     However, I don't see the names of the individuals who were killed. I will need to
     search further to find the specific individuals involved in the accident.

<search> Amtrak derailment 2015 victims </search>

<information>Doc 1(Title: "2015 Philadelphia train derailment") 2015 Philadelphia train
     derailment On May 12, 2015, an Amtrak ""Northeast Regional"" train from Washington,
     D.C. bound for New York City derailed and wrecked on the Northeast Corridor in the
     Port Richmond neighborhood of Philadelphia, Pennsylvania. Of 238 passengers and 5
     crew on board, 8 were killed and over 200 injured, 11 critically. The train was
     traveling at in a zone of curved tracks when it derailed. Some of the passengers
     had to be extricated from the wrecked cars. Many of the passengers and local
     residents helped first responders during the rescue operation. Five local hospitals
     treated the injured. The
Doc 2(Title: "2015 Philadelphia train derailment") 6, 2018, following an appeal of
     Gehret's ruling by the Pennsylvania Attorney General's Office, Philadelphia Common
     Pleas Court Judge Kathryn Streeter Lewis reinstated all criminal charges against
     Bostian. 2015 Philadelphia train derailment On May 12, 2015, an Amtrak ""Northeast
     Regional"" train from Washington, D.C. bound for New York City derailed and wrecked
     on the Northeast Corridor in the Port Richmond neighborhood of Philadelphia,
     Pennsylvania. Of 238 passengers and 5 crew on board, 8 were killed and over 200
     injured, 11 critically. The train was traveling at in a zone of curved tracks when
     it derailed. Some of the passengers had
Doc 3(Title: "2015 Philadelphia train derailment") the curve at which the derailment
     occurred, and that also would have limited the train's speed entering the curve.
```

Shortly after the derailment, Amtrak completed ATC installation on the northbound track. The 2015 wreck was the deadliest on the Northeast Corridor since 1987, when 16 people died in a wreck near Baltimore. At about 9:10 p.m. (EDT) on Tuesday, May 12, 2015, Amtrak's northbound ""Northeast Regional"" 188 departed Philadelphia's 30th Street Station en route from Washington, D.C., to New York City. The train consisted of seven cars hauled by a year-old Amtrak Cities Sprinter (ACS)-64 locomotive, No. 601. The engineer</information>

From Doc 1, I see that 8 people were killed in the derailment, but I still don't have the names of the individuals. I will need to search for a more detailed list of the victims.

<search> names of the 8 people killed in the 2015 Amtrak derailment </search>

<information>Doc 1(Title: "2015 Philadelphia train derailment") 2015 Philadelphia train derailment On May 12, 2015, an Amtrak ""Northeast Regional"" train from Washington, D.C. bound for New York City derailed and wrecked on the Northeast Corridor in the Port Richmond neighborhood of Philadelphia, Pennsylvania. Of 238 passengers and 5 crew on board, 8 were killed and over 200 injured, 11 critically. The train was traveling at in a zone of curved tracks when it derailed. Some of the passengers had to be extricated from the wrecked cars. Many of the passengers and local residents helped first responders during the rescue operation. Five local hospitals treated the injured. The
Doc 2(Title: "2015 Philadelphia train derailment") the curve at which the derailment occurred, and that also would have limited the train's speed entering the curve. Shortly after the derailment, Amtrak completed ATC installation on the northbound track. The 2015 wreck was the deadliest on the Northeast Corridor since 1987, when 16 people died in a wreck near Baltimore. At about 9:10 p.m. (EDT) on Tuesday, May 12, 2015, Amtrak's northbound ""Northeast Regional"" 188 departed Philadelphia's 30th Street Station en route from Washington, D.C., to New York City. The train consisted of seven cars hauled by a year-old Amtrak Cities Sprinter (ACS)-64 locomotive, No. 601. The engineer
Doc 3(Title: "2015 Philadelphia train derailment") 6, 2018, following an appeal of Gehret's ruling by the Pennsylvania Attorney General's Office, Philadelphia Common Pleas Court Judge Kathryn Streeter Lewis reinstated all criminal charges against Bostian. 2015 Philadelphia train derailment On May 12, 2015, an Amtrak ""Northeast Regional"" train from Washington, D.C. bound for New York City derailed and wrecked on the Northeast Corridor in the Port Richmond neighborhood of Philadelphia, Pennsylvania. Of 238 passengers and 5 crew on board, 8 were killed and over 200 injured, 11 critically. The train was traveling at in a zone of curved tracks when it derailed. Some of the passengers had</information>

From Doc 1, I still don't have the names of the individuals. I will need to search for a more detailed list of the victims.

<search> 2015 Amtrak derailment victim list </search>

<information>Doc 1(Title: "2015 Philadelphia train derailment") 2015 Philadelphia train derailment On May 12, 2015, an Amtrak ""Northeast Regional"" train from Washington, D.C. bound for New York City derailed and wrecked on the Northeast Corridor in the Port Richmond neighborhood of Philadelphia, Pennsylvania. Of 238 passengers and 5 crew on board, 8 were killed and over 200 injured, 11 critically. The train was traveling at in a zone of curved tracks when it derailed. Some of the passengers had to be extricated from the wrecked cars. Many of the passengers and local residents helped first responders during the rescue operation. Five local hospitals treated the injured. The
Doc 2(Title: "2015 Philadelphia train derailment") 6, 2018, following an appeal of Gehret's ruling by the Pennsylvania Attorney General's Office, Philadelphia Common Pleas Court Judge Kathryn Streeter Lewis reinstated all criminal charges against Bostian. 2015 Philadelphia train derailment On May 12, 2015, an Amtrak ""Northeast Regional"" train from Washington, D.C. bound for New York City derailed and wrecked on the Northeast Corridor in the Port Richmond neighborhood of Philadelphia, Pennsylvania. Of 238 passengers and 5 crew on board, 8 were killed and over 200 injured, 11 critically. The train was traveling at in a zone of curved tracks when it derailed. Some of the passengers had

Doc 3(Title: "2015 Philadelphia train derailment") the curve at which the derailment occurred, and that also would have limited the train's speed entering the curve. Shortly after the derailment, Amtrak completed ATC installation on the northbound track. The 2015 wreck was the deadliest on the Northeast Corridor since 1987, when 16 people died in a wreck near Baltimore. At about 9:10 p.m. (EDT) on Tuesday, May 12, 2015, Amtrak's northbound ""Northeast Regional"" 188 departed Philadelphia's 30th Street Station en route from Washington, D.C., to New York City. The train consisted of seven cars hauled by a year-old Amtrak Cities Sprinter (ACS)-64 locomotive, No. 601. The engineer</information>

From Doc 1, I still don't have the names of the individuals. I will need to search for a more detailed list of the victims.

<search> 2015 Philadelphia train derailment victim list </search>

<information>Doc 1(Title: "2015 Philadelphia train derailment") 2015 Philadelphia train derailment On May 12, 2015, an Amtrak ""Northeast Regional"" train from Washington, D.C. bound for New York City derailed and wrecked on the Northeast Corridor in the Port Richmond neighborhood of Philadelphia, Pennsylvania. Of 238 passengers and 5 crew on board, 8 were killed and over 200 injured, 11 critically. The train was traveling at in a zone of curved tracks when it derailed. Some of the passengers had to be extricated from the wrecked cars. Many of the passengers and local residents helped first responders during the rescue operation. Five local hospitals treated the injured. The
Doc 2(Title: "2015 Philadelphia train derailment") 6, 2018, following an appeal of Gehret's ruling by the Pennsylvania Attorney General's Office, Philadelphia Common Pleas Court Judge Kathryn Streeter Lewis reinstated all criminal charges against Bostian. 2015 Philadelphia train derailment On May 12, 2015, an Amtrak ""Northeast Regional"" train from Washington, D.C. bound for New York City derailed and wrecked on the Northeast Corridor in the Port Richmond neighborhood of Philadelphia, Pennsylvania. Of 238 passengers and 5 crew on board, 8 were killed and over 200 injured, 11 critically. The train was traveling at in a zone of curved tracks when it derailed. Some of the passengers had
Doc 3(Title: "2015 Philadelphia train derailment") by the derailment in 2015. This amount is in response to over 125 cases filed against Amtrak by passengers and family members, but can also be made available to passengers who had not yet filed a lawsuit or settled with Amtrak. On May 9, 2017, Philadelphia prosecutors said Bostian, the train's engineer, would not face any criminal charges because there was no evidence that he had acted with criminal intent. On May 12, 2017, on the anniversary of the crash, Philadelphia Municipal Court Judge Marsha Neifield ordered prosecutors to file involuntary manslaughter and reckless endangerment charges against Bostian; the district</information>

From Doc 1, I still don't have the names of the individuals. I will need to search for a more detailed list of the victims.

<search> 2015 Philadelphia train derailment victim names </search>

SMEPO Output

To answer this question, I need to identify the American social entrepreneur who was one of the eight killed in the 2015 Amtrak train derailment from Washington, D.C. to New York City, and determine her age at the time of the incident.

First, I'll search for information about the Amtrak train derailment in 2015 involving a train from Washington, D.C. to New York City, including the list of victims killed in the incident.

<search> Find information about the 2015 Amtrak train derailment from Washington, D.C. to New York City, including the list of victims killed in the incident. </search>

<information>Doc 1(Title: "2015 Philadelphia train derailment") 2015 Philadelphia train derailment On May 12, 2015, an Amtrak ""Northeast Regional"" train from Washington, D.C. bound for New York City derailed and wrecked on the Northeast Corridor in the Port Richmond neighborhood of Philadelphia, Pennsylvania. Of 238 passengers and 5

crew on board, 8 were killed and over 200 injured, 11 critically. The train was traveling at in a zone of curved tracks when it derailed. Some of the passengers had to be extricated from the wrecked cars. Many of the passengers and local residents helped first responders during the rescue operation. Five local hospitals treated the injured. The
Doc 2(Title: "2015 Philadelphia train derailment") 6, 2018, following an appeal of Gehret's ruling by the Pennsylvania Attorney General's Office, Philadelphia Common Pleas Court Judge Kathryn Streeter Lewis reinstated all criminal charges against Bostian. 2015 Philadelphia train derailment On May 12, 2015, an Amtrak ""Northeast Regional"" train from Washington, D.C. bound for New York City derailed and wrecked on the Northeast Corridor in the Port Richmond neighborhood of Philadelphia, Pennsylvania. Of 238 passengers and 5 crew on board, 8 were killed and over 200 injured, 11 critically. The train was traveling at in a zone of curved tracks when it derailed. Some of the passengers had
Doc 3(Title: "2015 Philadelphia train derailment") the curve at which the derailment occurred, and that also would have limited the train's speed entering the curve. Shortly after the derailment, Amtrak completed ATC installation on the northbound track. The 2015 wreck was the deadliest on the Northeast Corridor since 1987, when 16 people died in a wreck near Baltimore. At about 9:10 p.m. (EDT) on Tuesday, May 12, 2015, Amtrak's northbound ""Northeast Regional"" 188 departed Philadelphia's 30th Street Station en route from Washington, D.C., to New York City. The train consisted of seven cars hauled by a year-old Amtrak Cities Sprinter (ACS)-64 locomotive, No. 601. The engineer</information>

From the information, I've found that the Amtrak train derailment occurred on May 12, 2015, and 8 people were killed, including a social entrepreneur. Next, I'll search for the name of the social entrepreneur and her age at the time of the incident.

<search> Find the name of the American social entrepreneur who was one of the eight killed in the 2015 Amtrak train derailment from Washington, D.C. to New York City, and her age at the time of the incident. </search>

<information>Doc 1(Title: "Rachel Jacobs") Rachel Jacobs Rachel Jacobs (October 3, 1975 -- May 12, 2015) was an American social entrepreneur and CEO of a tech company. She was killed at age 39 in the 2015 Philadelphia train derailment while commuting between her home in New York and the Philadelphia offices of ApprenNet, the educational technology company she had recently joined as CEO. Jacobs grew up in Huntington Woods, Michigan, the daughter of Gilda Jacobs, a former Michigan state senator. She was a 1993 graduate of Berkley High School, a 1997 graduate of Swarthmore College, and a 2002 graduate of Columbia Business School. Jacobs moved
Doc 2(Title: "Rachel Jacobs") Safety Board to prevent future tragedies in which a two-year-old child ""grows up without a mother."" Assigned to ride on the first train through Philadelphia after the derailment, columnist Ronnie Polaneczky wrote for ""The Philadelphia Inquirer"" that, ""The next time I sigh that I can't afford to fix the roof or haven't time to help a friend move across the country, I will try to remind myself that Rachel Jacobs would have given anything to still be here to indulge such petty worries."" Rachel Jacobs Rachel Jacobs (October 3, 1975 -- May 12, 2015) was an American social entrepreneur and
Doc 3(Title: "Melissa Helmbrecht") several awards, including the White House Building Healthy Communities and Healthy Youth Award, the CBS Everyday Hero Award, the Walt Disney World Dreamers and Doers Award, and the National Caring Award. She was also inducted into the Frederick Douglass Museum and Hall of Fame for Caring Americans and was named ""One of the Six Leading Social Entrepreneurs in America"" by Youth Service America. Melissa Helmbrecht Melissa Helmbrecht (1975) is a New Jersey based social entrepreneur and activist who over the course of her career works to solve social problems through nonprofit work. Over the course of her career she</information>

Based on the information, the American social entrepreneur who was one of the eight killed in the 2015 Amtrak train derailment from Washington, D.C. to New York City was Rachel Jacobs. She was 39 years old at the time of the incident.

<answer>39</answer>

Outcome

Expert: Correct **GRPO:** Wrong **SMEPO:** Correct

E Additional Training and Evaluation Details

E.1 Training Details

Training data. We conduct training across three domains: mathematical reasoning, code generation, and agentic search. For mathematical reasoning and code generation, we construct fixed 1k-example hard subsets from OpenThoughts-derived math and code training pools [31]. To construct these subsets, we run Qwen3-4B-Instruct with four rollouts per problem and select examples for which all four rollouts are incorrect. This filtering is used only to obtain difficult training instances with reliable room for improvement; after selection, the same subset is shared by all compared methods. Each retained example contains the original task prompt, the reference answer or verifier, and an optional expert trace.

For agentic search, we construct a fixed 1k-example subset from MiroVerse-HotpotQA [32]. We first filter for examples whose final answer can be reliably extracted, and then randomly sample 1k examples from the filtered pool. This filtering ensures that the reward function can be applied consistently. Each retained example contains the user question, the search-interaction format, the reference answer, and an optional expert search trajectory.

In each domain, full-expert-trace, prefix-trace, random-mask, and SMEPO variants are constructed from the same underlying examples and expert traces. Thus, compared variants differ only in the guidance field, while the original task prompt, reference answer, verifier, and number of training examples are held fixed.

Expert-trace preprocessing. Expert traces are preprocessed offline before training. For RL-based guided training, each example is converted into several guidance variants constructed from the same original expert trace. The full-expert-trace variant exposes the complete expert trace. The prefix-trace variant exposes only an initial prefix of the trace. The random-mask variant replaces randomly selected spans with mask placeholders while matching the masking budget of SMEPO whenever applicable. SMEPO applies domain-specific masking rules that remove reward-relevant content while preserving the surrounding reasoning, program, or interaction semantics.

For math, we mask numerical content in the expert trace. Specifically, we replace standalone numeric spans, including decimal numbers and other typical numeric spans, with a special `[NUMBER]` placeholder before appending the processed trace to the original problem prompt. The regular expression is designed not to mask digits that are part of step or case labels, such as `Step 1` and `Case 1`. This removes many intermediate and final numerical values that can directly reveal the answer, while preserving surrounding algebraic transformations, natural-language reasoning, and non-answer-relevant labels.

For code, we mask code blocks in the expert trace and replace the entire code body with a single `[CODE]` placeholder. For example, a Python fenced block keeps the opening fence and language tag, replaces the body with `[CODE]`, and keeps the closing fence. We apply this preprocessing to all available expert-trace fields. This removes directly executable expert solutions while retaining the surrounding natural-language reasoning and the information that a code solution is required.

For agentic search, we mask named-entities over the expert search trace. Concretely, we use spaCy [15], a standard named-entity recognition tool, to detect named entities such as people, organizations, and locations in the trace. In addition, we extract answer spans and terms explicitly emphasized with bold markup, since these spans often correspond to final-answer entities, intermediate entities, query pivots, or supporting-page titles. We then replace all exact same-name occurrences of the detected or extracted entity strings within the entire trace, including the first occurrence, with `[ENTITY]`. The masked trace is appended to the original prompt, preserving the multi-step search trajectory while hiding critical entity cues that would otherwise reveal the search path or final answer.

For SFT, we use complete expert traces as supervision targets. For SFT+RL experiments in math tasks, the SFT stage also follows this complete-trace supervision setup. All preprocessed guidance inputs are generated once and kept fixed throughout training.

Implementation framework. We implement the math and code SFT/RL pipelines in VeRL [52]. For SFT, we use VeRL's FSDP-based supervised fine-tuning trainer. For RLVR training, we use VeRL's standard components for rollout generation, reward computation, advantage estimation, and policy optimization, with vLLM as the rollout backend. For agentic search, we use the Search-R1 training framework [35], which is also built on a VeRL-style RL pipeline but includes its own search-interaction environment, trajectory generation logic, and task-specific reward interface. The SFT stage for agentic search is implemented with the same VeRL-based SFT pipeline as the other domains.

Importantly, within each domain, SMEPO does not modify the underlying SFT loss, RL optimizer, reward function, verifier, rollout backend, or rollout engine. All compared variants in the same domain therefore share the same training code path for the corresponding stage, distributed execution backend, reward computation, and evaluation interface. The only difference across full-expert-trace, prefix-trace, random-mask, and SMEPO variants is the form of expert guidance exposed to the policy. This design ensures that the observed differences are attributable to the guidance format rather than to changes in the training framework or optimization procedure.

Training objective and reward. All RL runs use a GRPO-style RLVR objective. Expert guidance is included only as part of the policy input and is not used directly in reward computation. For math and code, rewards are computed by task-specific verifiers through VeRL's data-source-specific `default_compute_score` interface. For math datasets, the corresponding data sources are dispatched to `math_reward.compute_score`, which extracts the model's final answer, applies answer normalization, and compares it with the reference answer [52]. For code datasets, the corresponding data sources are dispatched to a PrimeCode-style non-continuous binary verifier: a generated program receives reward 1 only if it passes all associated test cases, and 0 otherwise [52].

For agentic-search, we use the reward implementation from the Search-R1 training codebase, which is dispatched to `qa_em.compute_score_em`. This function extracts the final answer from the model output using the `<answer>...</answer>` format, normalizes both the prediction and the reference answer by lowercasing, removing punctuation and articles, and fixing whitespace, and then assigns reward 1 for exact match and 0 otherwise [35].

Hyperparameters and compute. We use three different configurations: one shared SFT configuration, one RL configuration for math and code, and one RL configuration for agentic search. Within each domain, base model, and training stage, all compared variants use the same data split, compute setup, prompt template, optimizer settings, rollout configuration when applicable, and reward function.

For SFT, all domains use the same VeRL-based supervised fine-tuning setup. Training is conducted with bfloat16 precision and FSDP on a single node with 8 H100 GPUs unless otherwise stated. Pure SFT runs do not require rollout generation.

For math and code RL, we use VeRL's RLVR pipeline with vLLM as the rollout backend. Training is conducted with bfloat16 precision and FSDP on a single node with 8 H100 GPUs. For RL-based comparisons, the reward function, rollout configuration, GRPO hyperparameters, and verifier are fixed across variants.

For agentic-search RL, we use the Search-R1 training setup with a GRPO-style objective and vLLM as the rollout backend. Training is conducted on a single node with 8 H100 GPUs. Compared with math and code RL, agentic-search RL additionally includes an interactive search environment, multi-turn trajectory generation, and a local retriever. Within agentic search, the compute setup, search backend, rollout configuration, reward function, GRPO hyperparameters, and data split are fixed across variants.

For pure RL and pure SFT runs, we train on the full training set for the corresponding domain. For SFT+RL runs in math tasks, we split the same training set into two disjoint halves: the first half is used for SFT, and the second half is used for the subsequent RL stage. This ensures that SFT+RL does not use more training examples than the corresponding single-stage baselines. The RL stage in SFT+RL is initialized from the resulting SFT checkpoint and uses the same RL configuration as the corresponding pure RL run. Table 8 summarizes the shared compute and data setup, while Table 9 reports the hyperparameters.

Setting	Math	Code	Search
Framework	VeRL	VeRL	Search-R1
Hardware	8 H100	8 H100	8 H100
Precision	bf16	bf16	bf16
Update backend	FSDP	FSDP	FSDP
Base model	Qwen3-8B-Base	Qwen3-8B-Base	Qwen3-8B-Base
Train examples	1,024	1,024	1,024
Training source	OpenThoughts	OpenThoughts	MiroVerse-HotpotQA
Expert source	DeepSeek-R1	DeepSeek-R1	MiroVerse-HotpotQA trajectories

Table 8: **Shared compute and data setup for the main training experiments.** Math and code examples are selected from OpenThoughts-derived training pools, with DeepSeek-R1 expert traces. Agentic-search examples and expert trajectories are selected from MiroVerse-HotpotQA.

Hyperparameter	SFT	Math/Code RL	Search RL
Framework	VeRL	VeRL	Search-R1
Backend	N/A	vLLM	vLLM
Reward/verifier	N/A	Task-specific	Exact-match
Remove padding	Enabled	Enabled	Disabled
Learning rate	1×10^{-5}	1×10^{-6}	8×10^{-7}
Warmup ratio	0.1	0.1	0.285
Weight decay	0.01	0.1	1×10^{-2}
Adam β_1, β_2	0.9, 0.95	0.9, 0.999	0.9, 0.999
Gradient clipping	1.0	1.0	1.0
LR scheduler	Cosine	Cosine	Constant
Training steps	1000	400	400
Train batch size	256	32	32
Rollouts per prompt	N/A	8	8
Max sequence length	16,384	N/A	N/A
Max prompt / response length	N/A	4,096 / 12,288	4,096 / 2,048
Max start / observation length	N/A	N/A	2,048 / 2,048
Max search turns	N/A	N/A	5
Retriever top-k	N/A	N/A	3
Advantage estimator	N/A	GRPO	GRPO
KL coefficient	N/A	0.0	0.02
KL loss type	N/A	N/A	Low-var KL
Policy clip ratio	N/A	0.0, 0.5	0.2
Entropy coefficient	N/A	0.0	0.001
Rollout temperature	N/A	0.8	0.8
Rollout top-p	N/A	1.0	0.95

Table 9: **Hyperparameters for SFT, math/code RL, and agentic-search RL.** Pure SFT uses only the SFT configuration. Math and code RL use the same VeRL RLVR configuration. Search RL uses the Search-R1 setup with vLLM rollout, multi-turn search interaction, and a local retriever. SFT+RL first performs SFT on half of the training set and then runs RL on the other half, initialized from the SFT checkpoint.

E.2 Evaluation Details

We evaluate each benchmark with its corresponding verifier or official evaluation pipeline, and additionally report MMLU as a general-capability diagnostic.

Math evaluation. For mathematical reasoning, we evaluate on six benchmarks: GSM8K [19], MATH500 [20], AIME25 [21], AIME26 [22], AMC [23], and OlympiadBench [24]. We use a verifier adapted from lm-evaluation-harness [33]: the evaluator extracts the model's final answer, applies answer normalization, and compares it with the reference answer. For AIME25 and AIME26, each problem is repeated 32 times in the evaluation set; we generate one response for each repeated instance, score each response independently, and report the overall accuracy over all repeated instances. We report accuracy on each benchmark and the unweighted average across the six benchmarks.

Math							
Method	GSM8K	MATH500	AIME25	AIME26	AMC	Olympiad	Avg.
Qwen3-8B-Base							
GRPO	91.5 ± 0.5	74.8 ± 0.5	12.7 ± 0.4	11.8 ± 0.5	61.7 ± 2.9	28.9 ± 0.6	46.9 ± 0.7
Full Expert Trace	75.1 ± 1.4	69.7 ± 0.4	11.5 ± 0.2	9.0 ± 0.4	50.0 ± 2.5	27.5 ± 0.3	40.5 ± 0.4
SMEPO	92.3 ± 0.3	76.5 ± 0.4	16.8 ± 0.4	13.0 ± 0.3	68.3 ± 2.5	31.4 ± 0.4	49.7 ± 0.4
Qwen2.5-7B							
GRPO	88.7 ± 1.0	69.0 ± 1.1	8.7 ± 0.9	7.5 ± 0.2	54.2 ± 1.4	25.0 ± 0.6	42.2 ± 0.3
Full Expert Trace	83.6 ± 1.5	63.5 ± 0.8	5.6 ± 0.5	6.2 ± 0.2	46.7 ± 1.4	22.6 ± 0.8	38.0 ± 0.5
SMEPO	88.5 ± 0.7	72.3 ± 0.6	10.6 ± 0.6	9.5 ± 0.2	62.5 ± 2.5	26.0 ± 0.8	44.9 ± 0.3
DeepSeek-R1-Distill-Qwen-7B							
GRPO	87.6 ± 1.3	84.4 ± 0.5	34.2 ± 0.8	38.8 ± 0.3	88.3 ± 2.9	37.4 ± 0.6	61.8 ± 0.7
Full Expert Trace	87.3 ± 0.8	81.9 ± 1.4	34.6 ± 0.5	38.2 ± 1.0	83.3 ± 2.9	37.4 ± 0.7	60.5 ± 0.5
SMEPO	90.1 ± 2.4	86.3 ± 0.8	35.4 ± 0.8	41.6 ± 2.2	90.8 ± 1.4	39.2 ± 1.3	63.9 ± 1.5

Code: Qwen3-8B-Base				
Method	HumanEval	HumanEval+	LCB	Avg.
GRPO	66.7 ± 0.9	59.8 ± 1.3	41.3 ± 1.6	55.9 ± 1.1
Full Expert Trace	66.7 ± 0.3	58.1 ± 0.7	41.2 ± 1.0	55.3 ± 0.3
SMEPO	69.3 ± 0.9	62.2 ± 0.6	44.8 ± 0.7	58.8 ± 0.7

Agent: Qwen3-8B-Base	
Method	Bamboogle
GRPO	45.3 ± 0.5
Full Expert Trace	24.3 ± 1.2
SMEPO	49.1 ± 0.5

Table 10: **Three-seed results.** Math (top), code generation (bottom left), and agentic search (bottom right). Values are mean $\pm$ standard deviation across three independent training seeds. Math experiments use DeepSeek-R1 expert traces.

Domain	Masked tokens	Precision ↑	Recall ↑	Over-masking ↓	Under-masking ↓
Math	19.5%	89.0%	97.9%	11.0%	2.1%
Search	30.2%	80.6%	91.8%	19.4%	8.2%
Code	41.4%	100.0%	100.0%	0.0%	0.0%

Table 11: Quantitative masking quality across three domains.

Code evaluation. For code generation, we evaluate on HumanEval and HumanEval+ using lm-evaluation-harness with the vLLM backend [33]. We additionally evaluate on LiveCodeBench with the official LiveCodeBench runner [27] under the code-generation scenario. Both evaluation pipelines use execution-based pass@1 scoring: a generated program is counted as correct only if it passes the corresponding test cases. For the main results in Section 3, we use the `release_v1` split of LiveCodeBench.

Agentic-search evaluation. For agentic-search tasks, we evaluate on Bamboogle [28] using the Search-R1 evaluation pipeline, which uses a normalized exact-match metric. The model generates its own multi-turn search trajectory using the same search-interaction format as training. The final answer is extracted from the model output and scored by normalized exact match against the reference answer, using the same answer-normalization convention as the Search-R1 reward function.

General-capability evaluation. To check whether domain-specific training harms general ability, we additionally evaluate all models on MMLU [29, 30] using lm-evaluation-harness [33]. We use the 5-shot log-likelihood setting and report the aggregate MMLU score. This evaluation is separate from the task-specific verifier evaluations above and serves as a general-capability diagnostic.

F Multiple Training Seeds

We repeat the comparisons using three independent training seeds across five settings. Table 10 reports mean $\pm$ standard deviation. SMEPO achieves the best average in all five settings.

G Masking Coverage and Quality

We use GPT-5.4 to audit semantic-span masking quality over the full datasets in all three domains, and manually validate 50 randomly sampled traces per domain. Metrics are computed at the semantic-span level, with each maximal masked or missed span counted as one unit. Precision measures

Domain	Agreement	Confirmed Claims	Additional Missed Spans
Math	97.5%	269/276	0
Search	98.8%	165/167	3
Code	100.0%	50/50	0

Table 12: Manual validation of masking quality on 50 randomly sampled traces per domain.

Variant	Masked	GSM8K	MATH500	AIME25	AIME26	AMC	Olympiad	Avg.
Drop20	15.6%	90.4	75.0	15.2	10.4	62.5	30.7	47.4
Drop10	17.6%	91.7	76.2	16.7	12.5	65.0	30.4	48.8
SMEPO	19.5%	92.2	76.6	17.6	13.4	70.0	32.0	50.3
Add10	27.6%	91.6	76.8	17.2	13.2	70.0	31.8	50.1
Add20	35.6%	89.6	76.8	15.6	12.5	65.0	30.3	48.3

Table 13: Robustness to perturbations of math masking density on Qwen3-8B-Base. "Masked" is the resulting token fraction.

whether masked spans are reward-relevant, and recall measures whether all reward-relevant spans are masked. The results are reported in Table 11.

Manual validation shows close agreement with the automatic audit, as summarized in Table 12.

Overall, the lightweight masking rules already identify reward-relevant spans with high reliability. Recall reaches 97.9% for Math, 91.8% for Search, and 100.0% for Code, while precision remains 89.0%, 80.6%, and 100.0%, respectively. Moreover, the automatic audit closely matches human inspection, with 97.5% agreement in Math and 98.8% in Search. This supports the reliability of the span detector under our audit criterion and the large-scale masking-quality measurements.

H Robustness to Masking Density

To test whether SMEPO requires nearly perfect rules, we also perturb the masking extent over a wide range in the Math setting:

- **Reduced masking (drop10/drop20):** We randomly restore 10% or 20% of the tokens originally masked by SMEPO.
- **Additional masking (add10/add20):** We randomly mask an additional 10% or 20% of the tokens that remain unmasked under SMEPO, i.e., from the remaining 80.5% of the corpus.

As shown in Table 13, performance remains strong across a broad masking range. Moderate additional masking has almost no effect, while restoring masked content causes a larger drop, showing that SMEPO is robust to realistic masking imperfections and benefits from prioritizing high recall of leakage-bearing spans.

I Compatibility with GHPO

As shown in Table 2, SMEPO improves GHPO in Math, Code, and Agentic Search. Since GHPO does not natively support Code or multi-round Search, we adapt its original configuration and compare GHPO with GHPO+SMEPO under identical settings within each domain.

J Limitations and Future Work

SMEPO is designed as a general masked-guidance principle for RLVR settings where expert traces may expose outcome-informative content, including final answers and intermediate spans along the critical path to obtaining them. Due to computational and engineering constraints, our empirical study focuses on three representative domains: mathematical reasoning, code generation, and search-based agents. These domains cover different forms of outcome-informative content, including numerical values, executable programs, and answer-relevant entities, but they do not exhaust the full space of RLVR tasks and the corresponding verifier designs.

A natural direction for future work is to evaluate SMEPO on broader and evolving RLVR benchmarks, and to develop more automatic ways to identify outcome-informative spans under new task formats and verifier semantics. This may involve combining lightweight heuristics with stronger semantic parsers or learned span detectors, especially as RLVR evaluation increasingly includes more complex agentic tasks.

K Broader Impacts

This work studies a training method for expert-guided RLVR. Its main potential benefit is to improve the effectiveness of expert-guided RL training by reducing trace-induced reward hacking: SMEPO masks reward-informative spans so that the policy is less likely to exploit exposed answers, executable programs, or relevant entities.

The potential negative impacts are indirect. More capable reasoning and code-generation models could be misused in similar ways to other generative models, such as producing incorrect but persuasive outputs or assisting unsafe code generation. However, this paper does not release a deployed system, a new high-risk model, or a dataset designed for harmful use; our experiments are conducted on standard academic benchmarks. Models trained with SMEPO should still undergo standard safety and robustness evaluations before deployment, especially in settings involving code execution, tool use, or external information access.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes].

   Justification: The abstract and introduction state the main claims: full expert-trace conditioning can hurt RLVR, SMEPO masks reward-relevant content while preserving trace structure, and the method improves over vanilla GRPO and full-trace conditioning across math, code, and agentic search. The claims are supported by the main results, diagnostic table, training-efficiency analysis, GHPO compatibility study, and ablations.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes].

   Justification: Appendix J discusses the limitations.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A].

Justification: The paper is empirical and methodological, and does not present formal theoretical results, theorems, or proofs.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes].

Justification: The paper describes the evaluated models, datasets, expert-trace sources, compared methods, evaluation benchmarks, training configuration, rollout settings, reward computation, and hardware setup in the experiments section and appendix.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in

some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes].

   Justification: We publicly release the code, data preprocessing scripts, training and evaluation configurations, and instructions for reproducing the main experimental results. The repository is linked in the paper.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes].

   Justification: The experiments section and appendix specify the models, datasets, expert-trace sources, evaluation benchmarks, rollout count, maximum response length, learning rate, training steps, hardware, and evaluation protocols. Additional training and evaluation details are provided in the appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes].

   Justification: We report mean and standard deviation across three independent training seeds for five experimental settings, covering three math backbones, code generation, and agentic search (Appendix F). SMEPO consistently achieves the highest average performance across all five settings.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes].

   Justification: The paper reports that RL training uses 8×H100 GPUs for the main experiments and provides training steps and wall-clock target-reaching times. Additional compute and evaluation details are included in the appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes].

   Justification: The work studies training methods for RLVR using existing public benchmarks, open-weight or publicly accessible models, and standard evaluation datasets. We do not conduct human-subject experiments, collect private data, or release high-risk models or datasets.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes].

    Justification: The paper includes a broader impacts discussion in Appendix K.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A].

    Justification: The paper does not release a new pretrained language model, image generator, or scraped dataset with high misuse risk. The work uses existing models and benchmarks to study a training method.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes].

Justification: The paper cites the models, datasets, and benchmarks used in the experiments, including the base models, expert-trace sources, math/code/search benchmarks, and evaluation tools. We follow the access and usage terms of the corresponding assets.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [N/A].

Justification: The paper does not introduce a new public dataset, benchmark, or model asset as its primary contribution. The contribution is a training method and its empirical evaluation.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A].

Justification: The paper does not involve crowdsourcing experiments or research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A].

    Justification: The paper does not involve human-subject research, so IRB approval or equivalent review is not applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes].

    Justification: The paper uses LLMs as base policies and expert models for generating expert traces, which are central to the method and experiments. The models and trace sources are described in the experiments section.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.