Distilling LLM Feedback for Lean Theorem Proving

Gaëtan Narozniak
FAIR at Meta, Inria
gaetan@meta.com

Gérard Biau
Sorbonne Université,
Institut universitaire de France

Rémi Munos
FAIR at Meta

Ahmad Rammal
FAIR at Meta,
CERMICS École des Ponts ParisTech

Pierre Marion
Inria, ENS, PSL Research University
pierre.marion@inria.fr

Abstract

Post-training for reasoning models typically combines supervised fine-tuning with reinforcement learning from verifiable rewards, most commonly with GRPO. However, this algorithm suffers from sparse rewards, limited exploration, and mode collapse. Building upon recent works on self-distillation, we propose *Feedback Distillation*, a training method where the model is trained to match, at the token level, its own distribution conditioned on privileged feedback produced by a language model. Feedback Distillation is on-policy, offers token-level supervision, and can inject external knowledge. Evaluating our method for Lean 4 theorem-proving, we find that Feedback Distillation maintains greater diversity in generated trajectories than GRPO, yielding higher policy entropy and better pass@k scaling. The two methods are complementary: initializing GRPO from a Feedback Distillation checkpoint outperforms either method alone. All in all, our results suggest a promising avenue to improve post-training for complex reasoning.

1 Introduction

Recent advances in AI have demonstrated remarkable progress in complex reasoning tasks (Li et al., 2022; Rozière et al., 2023; Yao et al., 2023; Besta et al., 2024; Rein et al., 2024). Mathematical reasoning has emerged as a particularly valuable testbed for these approaches due to its challenging search spaces, clear correctness criteria, and rich problem structure ranging from formal theorem proving to informal problem-solving (Lightman et al., 2023; Trinh et al., 2024; Hubert et al., 2025; Peyronnet et al., 2026). A fruitful strategy uses verifiable rewards, where deterministic verifiers (e.g., Lean proof checkers) evaluate model outputs. In this setting, state-of-the-art post-training pipelines combine supervised fine-tuning (SFT) on expert solutions (coming either from human experts or from a stronger LLM) with reinforcement learning (RL) to maximize the reward provided by the verifier (Kumar et al., 2025). The RL algorithm is usually GRPO (Shao et al., 2024), an algorithm that samples multiple outputs per prompt and optimizes the policy by contrasting their rewards.

However, this SFT+GRPO paradigm has significant limitations. First, the SFT phase is limited by the difficulty and cost of generating the fine-tuning dataset. Furthermore, it is inherently *off-policy*: the model is trained on externally generated demonstrations rather than on trajectories sampled from its own policy, a distribution mismatch that can cause catastrophic forgetting outside the fine-tuning distribution (see, e.g., Ross and Bagnell, 2010; Agarwal et al., 2024; Shenfeld et al., 2026b,a). Moreover, recent evidence suggests GRPO does not amplify the diversity of model answers nor favor exploration during training, but rather collapses onto modes with high reward already present before RL training (Yue et al., 2025). This is particularly problematic in formal mathematics where exploration of large libraries of lemmas (e.g., Lean's Mathlib, The mathlib Community, 2020) is

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

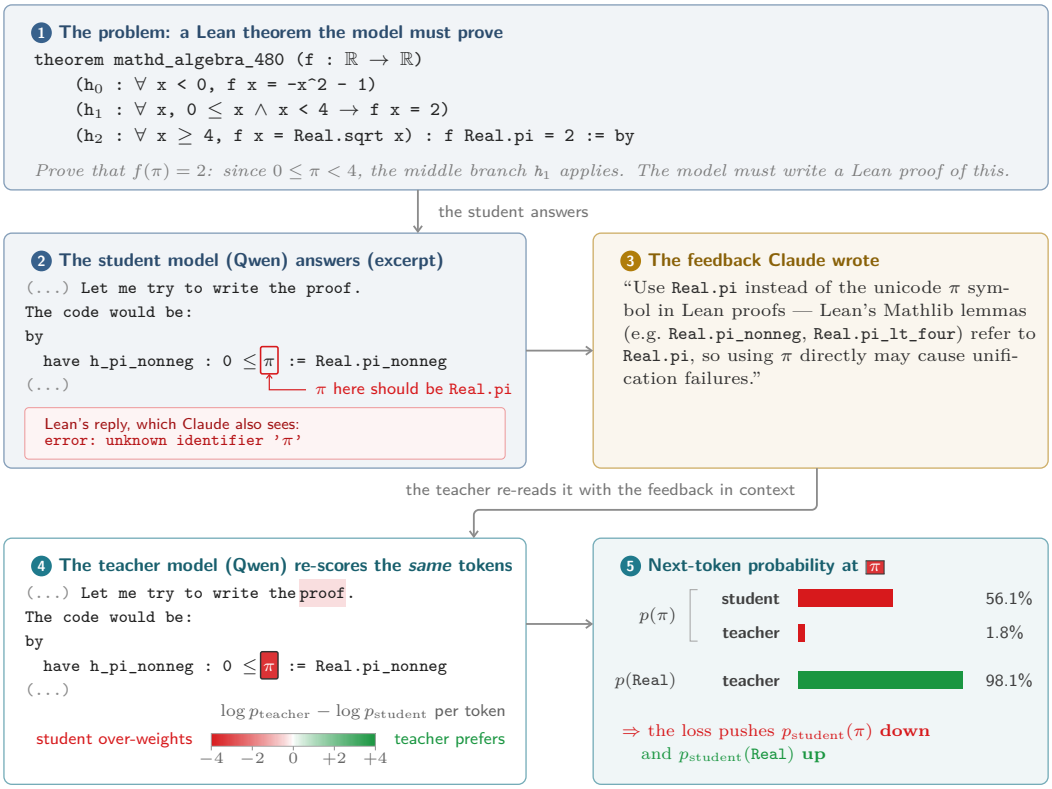

Figure 1: Example of Feedback Distillation on a Lean statement from the MiniF2F dataset (Zheng et al., 2022), with Qwen3-8B as the student. Claude provides feedback on the student's attempt. The student writes "π" where the teacher would have written "`Real`" with 98% probability, reflecting Claude's feedback. Through the KL loss, the student is trained to increase the probability of `Real` at this position, and thus learns to write better Lean code.

essential for success. Finally, GRPO provides only trajectory-level rewards, which offer poor credit assignment, and no learning signal when all samples fail.

These interconnected challenges motivate exploration of alternative training mechanisms. A promising approach is the recently proposed self-distillation algorithm (Hübotter et al., 2026; Shenfeld et al., 2026a; Zhao et al., 2026), which trains the model to mimic its own outputs when conditioned on privileged information, distilling this information into the model's weights. This approach is attractive for two distinct reasons: the trajectories it trains on are sampled from the current policy, avoiding the distribution mismatch inherent to off-policy SFT; and the feedback-conditioned distribution is matched at the token level, providing fine-grained credit assignment rather than a reward distributed uniformly across entire trajectories (see example in Figure 1). It thus combines the on-policy nature of GRPO with a supervision signal denser than that of SFT. Moreover, it can apply beyond verifiable reward settings since it does not directly rely on verifier signals.

On-policy adaptation to a new environment. Proving theorems in Lean 4 is a capability our base models largely lack, so we view training on it as adapting the model to a new environment. Our focus is on whether that new ability can be acquired *on-policy*, which has been shown to enable better continual learning and to preserve existing capabilities more effectively than off-policy training such as SFT (Agarwal et al., 2024; Shenfeld et al., 2026b,a). Using a lagged version of the student's weights to update the teacher creates a feedback loop between the two models, which plays a crucial role in stabilizing the training process and getting a good performance.

Contributions. In this work, we investigate the performance of self-distillation in the context of training LLMs for mathematics. More precisely,

- We propose *Feedback Distillation*, a variant of self-distillation in which the privileged information is feedback produced by a third-party LLM or a frozen copy of the model in response to the model's attempt and the verifier's output, rather than raw environment signals or ground-truth solutions (Section 3).
- We evaluate Feedback Distillation in the Lean 4 formal mathematics environment (Section 4). We show that it maintains higher policy entropy than GRPO and achieves better pass@k scaling, indicating greater output diversity. Furthermore, we show that the two methods are complementary: initializing GRPO from a Feedback Distillation checkpoint outperforms GRPO alone. After training on LeanWorkbook, Qwen3-8B achieves 23% pass@1 test accuracy on MiniF2F with GRPO alone compared to 31% with Feedback Distillation+GRPO.
- We provide a detailed analysis of Feedback Distillation: we compare with other sources of feedback, and discuss training instability and qualitative consequences of our training methodology, which can be seen as a form of on-policy knowledge transfer (Section 5).

2 Related work

Self-distillation for LLMs. A recent line of work trains LLMs by minimizing the divergence between a student and a teacher that shares the same architecture but is conditioned on privileged information. Hübotter et al. (2026) use environment outputs as the privileged signal while Shenfeld et al. (2026a); Zhao et al. (2026) use a ground-truth solution as the privileged information. Our work extends this line of research by using feedback from a frozen copy of the model or from a third-party LLM. Furthermore, we perform a careful analysis of the properties of self-distillation, demonstrating in particular that it maintains greater trajectory diversity than GRPO, which translates into better test-time scaling when starting from a weak student in a difficult RL environment (formal mathematics).

RL for formal mathematics. State-of-the-art provers in formal mathematics usually follow a two-phase pipeline (Lin et al., 2025; Ren et al., 2025): supervised fine-tuning on synthetic proof data coming from the formalization of natural-language proofs, followed by GRPO-based RL, often referred to as reinforcement learning with verifiable rewards (RLVR). Recent progress on benchmarks such as MiniF2F (Zheng et al., 2022) and PutnamBench (Tsoukalas et al., 2024) has increasingly come from test-time scaling rather than improved training (e.g., Varambally et al., 2026). This scaling is achieved through agentic pipelines combining a trained prover with a reasoning model and iterative interaction with the Lean environment. While these scaffolding strategies are orthogonal to our work, they underscore the continued importance of strong base provers, whose training we aim to improve.

3 Feedback Distillation

Following the self-distillation methodology (Hübotter et al., 2026; Shenfeld et al., 2026a; Zhao et al., 2026), our training loss measures the divergence between two instances of the same model with independent weights, a *student* and a *teacher*, the latter receiving privileged information in its prompt. Both instances are initialized from the same pretrained LLM π. We seek to post-train on a task for which we have a dataset D of problems without solutions (e.g., Lean statements without proofs).

Specifically, given an attempted proof $y \sim \pi_\theta(\cdot \mid x)$ for a problem $x \sim D$, the teacher is given by $\pi_\mu(\cdot \mid x,\, F(y))$ where F is a function that takes an attempted response y and returns a textual feedback signal (the privileged information). For consistency with prior work, we refer to it as the teacher model, though it would be more accurate to describe it as an augmented student.

In our setting, the feedback F is an LLM that analyzes the attempt y. Since the feedback may originate from a stronger model, we use the term *Feedback Distillation*. The feedback model may also be a frozen copy of π, which we sometimes refer to as Self-Feedback Distillation. We refer to Sections 2 and 5.1 for discussion on other sources of feedback. We give examples of feedback in Figure 1 and Appendix D.

Following Hübotter et al. (2026), we update the teacher's weights via exponential moving average (EMA): every five gradient steps, we set $\mu \leftarrow \alpha\,\mu + (1-\alpha)\,\theta$, where $\alpha \in [0, 1]$ is a hyperparameter. This hyperparameter has a critical impact on stability and performance (see Section 5.2).

Our loss is the per-token Kullback-Leibler divergence between the teacher and student distributions along trajectories sampled from the student, namely:

$$\mathcal{L}_\theta \;=\; \mathbb{E}_{x\sim D,\; y\sim\pi_\theta(\cdot|x)}\left[\sum_{t=1}^{|y|}\mathrm{KL}(\pi_\mu(\cdot \mid x, F(y), y_{<t}) \;\|\; \pi_\theta(\cdot \mid x, y_{<t}))\right],$$

where gradients are not propagated through the sampling of y, and $|y|$ denotes the length of the sequence y. With V the vocabulary, the per-token KL divergence is

$$\mathrm{KL}(\pi_\mu(\cdot \mid x, F(y), y_{<t}) \;\|\; \pi_\theta(\cdot \mid x, y_{<t})) \;=\; \sum_{a\in V}\pi_\mu(a \mid x, F(y), y_{<t})\log\frac{\pi_\mu(a \mid x, F(y), y_{<t})}{\pi_\theta(a \mid x, y_{<t})}.$$

Since the teacher's distribution does not depend on θ, the KL divergence and the cross-entropy have the same gradient with respect to θ (see Appendix B). For a sampled problem $x \sim D$ and attempt $y \sim \pi_\theta(\cdot \mid x)$ with feedback $F(y)$, this yields the following equivalent loss, which we use in practice:

$$\hat{\mathcal{L}}_\theta \;=\; -\sum_{t=1}^{|y|}\sum_{a\in V}\pi_\mu(a \mid x,\, F(y),\, y_{<t})\;\log\pi_\theta(a \mid x,\, y_{<t}).$$

Top-K truncation. To reduce computational cost, we truncate the sum over the vocabulary to the K tokens with the highest probability under the teacher $\pi'_{\mu,y}$, using $K = 25$ in practice.

4 Capabilities of Feedback Distillation for Lean Theorem Proving

In this section, we provide evidence that training with Feedback Distillation enables the model to learn efficiently, outperforming GRPO in performance and policy diversity.

4.1 Lean environment

Our experimental setting is the Lean 4 formal mathematics environment (de Moura and Ullrich, 2021). It is a well-suited testbed for studying RL for complex reasoning for several reasons: (i) our base models (Qwen3-8B, Team, 2025, and Qwen3.5-9B, Team, 2026) have little proficiency in Lean 4, scoring respectively 2% and 11% on MiniF2F at initialization in our setup, so any meaningful improvement requires discovering new capabilities; (ii) the search space is vast, as the model must learn to invoke appropriate tactics and reference lemmas from Mathlib, Lean's main mathematical library comprising over 300,000 declarations; (iii) the Lean verifier provides ground-truth feedback on whether a proof attempt is correct. Results in this section are given for Qwen3-8B, and for Qwen3.5-9B in Section 5.

Tool interface. Tool use is a key component of state-of-the-art formal mathematics models (Liu et al., 2026), enabling iterative interaction with the proof assistant rather than single-shot generation. When the model produces a tool call, generation stops, the tool is executed, and its output is appended to the context before resuming generation. During training, the loss is computed only on tokens generated by the model and not on tokens produced by the tools. We equip the model with three tools to interact with the Lean environment.

- `lean_write_file`: write a Lean source file;
- `lean_check_file`: compile a previously written file and return the compiler output (errors, warnings, or success);
- `rg_in_mathlib`: search Mathlib using regular expressions to find relevant lemmas.

We confirm in Appendix A.2 that tool use improves performance in our setting. A problem is considered solved when the model writes a Lean file that compiles and proves the statement.

4.2 Experimental setting

We describe here the main ingredients of the experimental setting. Hyperparameters used can be found in Appendix A.1.

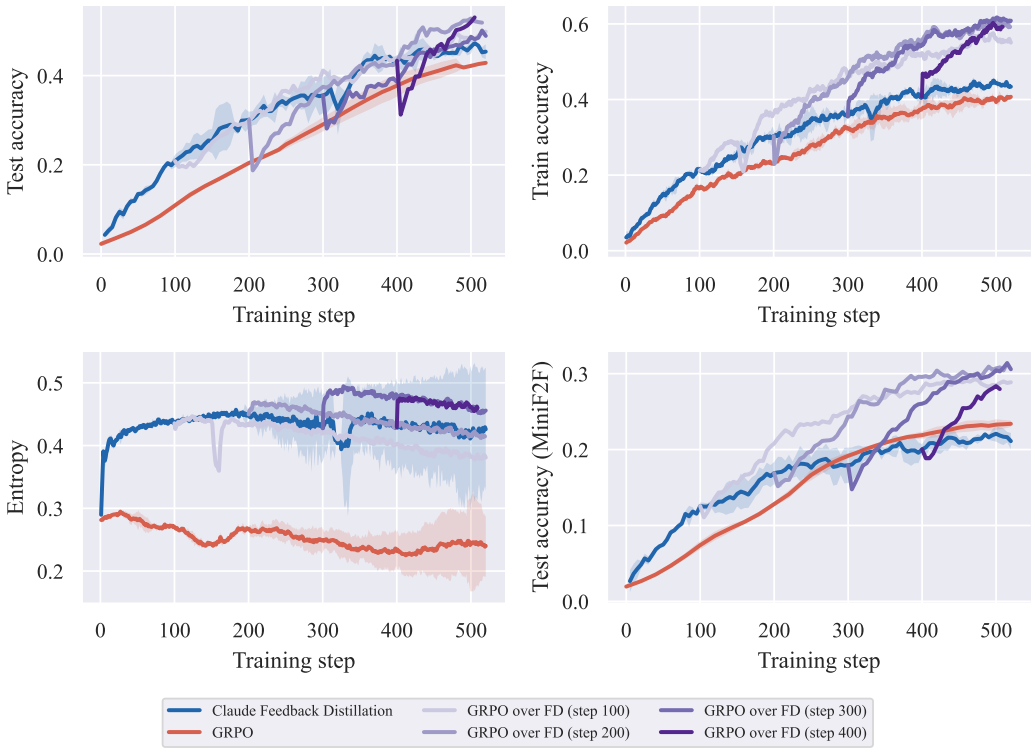

Figure 2: **Overview on Qwen3-8B.** LeanWorkbook test accuracy, train accuracy, policy entropy, and MiniF2F test accuracy for Claude Feedback Distillation, GRPO, and GRPO initialized from Feedback Distillation checkpoints at 100, 200, 300 and 400 steps. Curves are means over two seeds for GRPO and Claude Feedback Distillation, and shaded bands span ± 1 standard deviation. GRPO over Feedback Distillation reaches the highest accuracy, suggesting that Feedback Distillation provides a good initialization for GRPO.

Datasets and generation. We train on a subset of 10,000 statements with known proofs drawn from LeanWorkbook (Ying et al., 2025). For evaluation, we use a test set of LeanWorkbook comprising 256 statements as well as the test split of MiniF2F (Zheng et al., 2022), a standard benchmark of 244 competition-level problems formalized in Lean. Each attempt consists of a single generation of up to 8,192 tokens, within which the model may invoke multiple tool calls. All experiments are run on a single node equipped with 8× NVIDIA H200 GPUs. GRPO experiments take on average 550s per training step, while Feedback Distillation requires 300s per step. We use the `verl` codebase for the experiments (Sheng et al., 2024) and the Kimina Lean Server (Santos et al., 2025) to assess the correctness of Lean proofs.

Feedback model. We experiment with Claude Opus 4.6 (Anthropic, 2026) as the feedback model (see Section 5.1 for discussion on sources of feedback). The feedback model receives the formal statement x, the model's attempt y (including all tool calls and their outputs), and the final Lean compiler output. It is prompted to produce a few actionable directives of the form *"Do X"* or *"Don't do X"*, identifying proof strategy errors, flagging misuse of tactics or lemmas, or encouraging better tool usage (see Appendix A.3). The feedback is framed to be useful for a new attempt at the same problem. Examples of feedback are shown in Appendix D.

4.3 Results

We compare Feedback Distillation with GRPO, and also experiment with *GRPO over Feedback Distillation*, meaning that we take a checkpoint from Feedback Distillation and continue training with GRPO. Figure 2 gives an overview of this comparison for Qwen3-8B. We observe that GRPO over Feedback Distillation reaches the highest accuracy, while Feedback Distillation maintains a

policy entropy above that of GRPO. Feedback Distillation is more compute efficient than GRPO, as it draws one sample per prompt where GRPO draws five; Table 2 in Appendix A.4 gives the full accounting. Overall, **Feedback Distillation is able to learn efficiently while maintaining diversity in the generated trajectories**, a desirable property in light of recent findings that GRPO tends to reduce the diversity of model outputs during training (Dang et al., 2025; Yue et al., 2025; Wu et al., 2026).

GRPO over Feedback Distillation performs best. We take checkpoints after 100, 200, 300, and 400 steps of Feedback Distillation and continue training with GRPO. As shown in Figure 2, this combination outperforms GRPO alone. This suggests that the two methods provide complementary benefits.

Number of discovered lemmas. A concrete manifestation of Feedback Distillation's greater trajectory diversity is the rate at which the model invokes new Mathlib lemmas in successful proofs over the course of training. As shown in Figure 3, Feedback Distillation leads to a faster and more sustained increase in the number of distinct lemmas invoked, compared to GRPO.

Entropy and pass@k. Feedback Distillation maintains significantly higher entropy than GRPO throughout training, indicating that the policy preserves more diversity (Figure 2). This translates into better test-time scaling: at checkpoints where GRPO and Feedback Distillation achieve similar pass@1, Feedback Distillation scales significantly better with increasing k (Figure 3), confirming that the higher entropy reflects answer diversity rather than uniform noise.

Sample efficiency. Feedback Distillation draws a single sample per prompt where GRPO draws five, so at equal step count it has consumed five times fewer rollouts (see Appendix A.1).

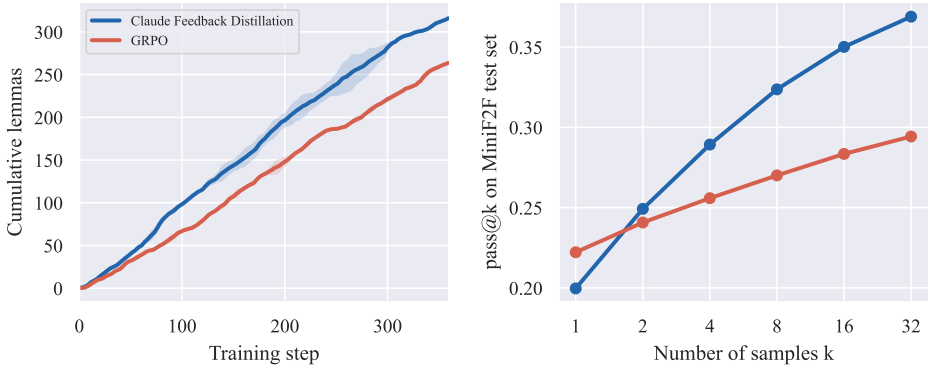

Figure 3: (Left) Discovery of Mathlib lemmas: cumulative number of distinct lemmas used in correct proofs over the course of training. Feedback Distillation leads to faster discovery of new lemmas from the Mathlib library. (Right) Pass@k scaling for two checkpoints at 400 steps of training with GRPO and Feedback Distillation (from the experiment in Figure 2), where both methods have similar pass@1 accuracy. Feedback Distillation scales better with k, confirming that its higher entropy translates into greater answer diversity.

5 Analysis of Feedback Distillation

This section provides further analysis of Feedback Distillation. We begin by discussing other sources of feedback than LLM-generated advice based on a prior attempt, then the key role of EMA and its connection to training instability, before some qualitative comments.

5.1 Other sources of feedback

The experiments in this subsection use Qwen3.5-9B, as it more consistently picked up training signal from the different feedback sources than Qwen3-8B. We observe training instabilities for GRPO over

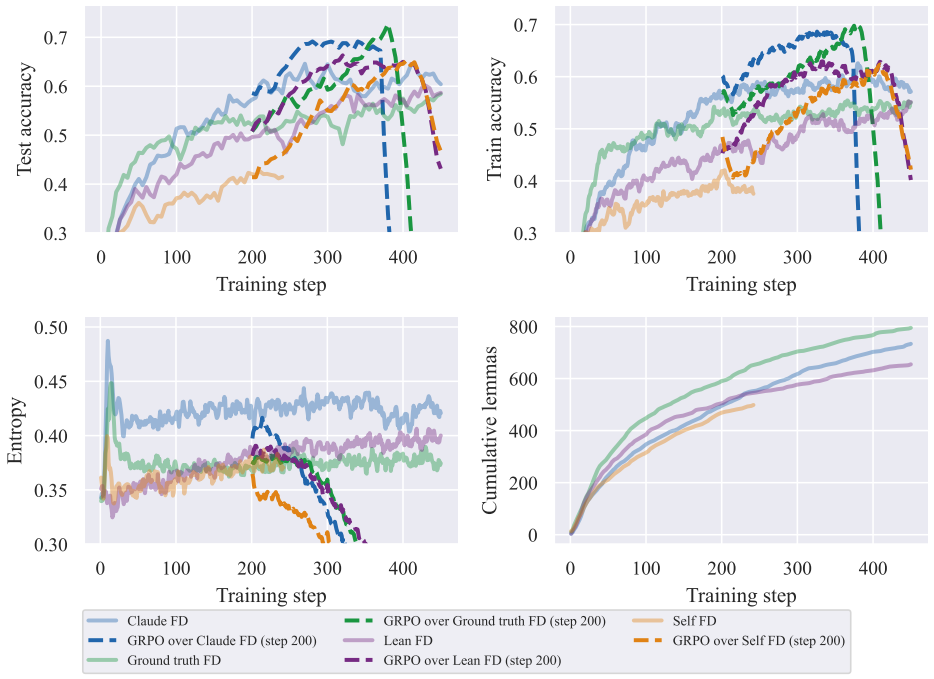

Figure 4: Feedback Distillation and GRPO over Feedback Distillation on Qwen3.5-9B for different feedback sources: Claude, a ground truth Lean solution, Lean compiler output, and Qwen itself (Self-Feedback Distillation). Claude and ground truth feedback sources are a better initialization for the GRPO training.

Feedback Distillation on Qwen3.5-9B. Similar instabilities were observed in our setup when training the base Qwen3.5-9B model with GRPO. We expect that these instabilities could be mitigated by hyperparameter tuning, and do not affect the qualitative conclusions we draw in this subsection. We consider three alternative feedback sources: a ground truth proof (Shenfeld et al., 2026a; Zhao et al., 2026), the raw Lean compiler output (Hübotter et al., 2026), and the trained model itself, which we discuss separately below. The ground truth feedback is a correct proof for the LeanWorkbook statement, formatted as ‘‘`Example of a correct solution:{solution}`’’. In case of failure, the Lean output feedback is the raw Lean compiler output formatted as ‘‘`The following is feedback from your unsuccessful earlier attempt: {environment_output}`’’, and in case of success it is the solution found, with the same format as for the ground truth feedback.

A comparison of the performance is presented in Figure 4. We observe that Claude and ground-truth as feedback sources outperform Lean output. This is expected, as both settings provide access to stronger sources of information. We hypothesize that the performance of Claude-based feedback could be further improved through better prompt design. In principle, it could even outperform both Lean output and ground truth, since Claude has access to compiler outputs and is sufficiently capable to generate correct solutions for LeanWorkbook problems.

Self-Feedback Distillation. We consider an alternative source of feedback in which the feedback model is not a stronger external model, but the same model being trained, with frozen weights. We refer to this setting as *Self-Feedback Distillation*. This setup does not introduce any additional source of information, and thus isolates the model's ability to improve based on its own analysis of its generations. The performance does not yet match that of the other feedback sources. We believe that further improvements may be achieved in this direction by refining the content and formulation of the feedback, pointing to promising directions for future work.

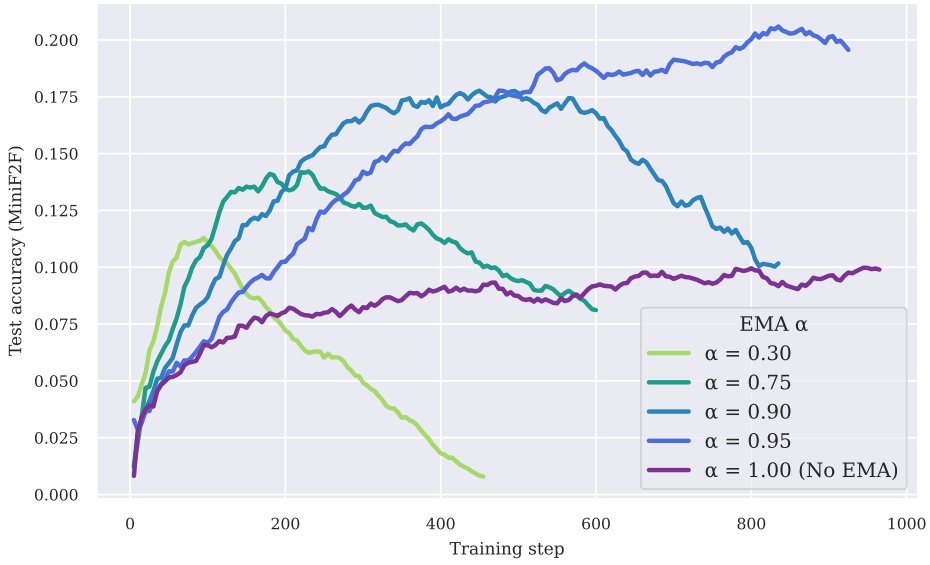

Figure 5: MiniF2F accuracy for Qwen3-8B during training for different values of the EMA interpolation parameter α. More aggressive EMA updates (i.e., lower values of α) accelerate early learning but result in increased instability. The case $\alpha = 1$ corresponds to freezing the teacher weights. We use $\alpha = 0.9$ for all the experiments.

5.2 EMA and training instability

Since the model learns from multiple feedback signals over the course of training, a frozen teacher would be problematic: knowledge acquired from earlier feedback would be penalized, since the teacher would not internalize this information and assign it low probability. EMA updates of the teacher address this issue by letting it progressively track the student (Section 3). This echoes self-supervised learning, where EMA target networks are widely used to prevent the representational collapse arising from moving targets (see, e.g., Grill et al., 2020; He et al., 2020; Assran et al., 2023). The EMA interpolation parameter $\alpha \in [0, 1]$ controls the tradeoff between adaptivity and stability. To assess this, we report in Figure 5 a sweep over α (see Appendix C.1 for the same sweep with Qwen3.5-9B). We find that more aggressive updates lead to faster learning early in training, while $\alpha = 1$ avoids instability at the cost of slow learning. Nevertheless, the long-term instability of training even with large α suggests stabilization as an important area for future work.

5.3 Qualitative comments on Feedback Distillation

We give below a few additional qualitative comments on Feedback Distillation.

The feedback is internalized into the model's weights. To verify this, we study a simplified setting where the feedback $F(y)$ is a fixed string throughout training. We consider two cases: "Be concise" or "Think carefully before answering. Verify that the solution is correct." In this experiment, the teacher is frozen. As shown in Figure 6, the average response length evolves consistently with the feedback, and the KL divergence between student and teacher vanishes. This confirms that the student learns to match the teacher's behavior by distilling the privileged information into its weights: **after training, the student behaves as if the feedback were concatenated to every prompt**.

Feedback formulation. The formulation of the prompt sent to the feedback model to generate feedback plays a key role in training performance. Increasing the level of structure and guidance in this prompt (e.g., enforcing formatting constraints and requiring self-contained feedback) improves stability and final performance, while slowing early learning. We refer to Appendix C.3 for a detailed analysis.

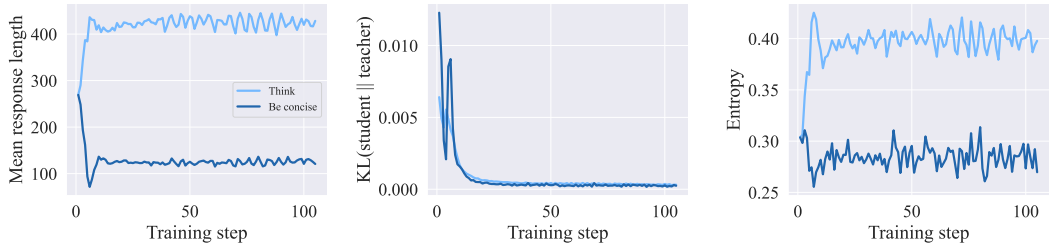

Figure 6: Effect of constant feedback on response length and policy entropy. "Think" feedback increases response length while "Be concise" decreases it. The KL divergence between student and teacher converges to zero, confirming that the feedback is internalized into the model's weights.

Epistemic verbalization. Kim et al. (2026a) show that self-distillation with rich privileged information (e.g., a ground-truth solution) can suppress epistemic verbalization, i.e., the model's use of uncertainty words such as `wait`, `hmm`, `perhaps`, leading to degraded out-of-distribution performance. We observe in Appendix C.4 that this suppression occurs significantly less often when the feedback is phrased as a critique, as in Feedback Distillation, suggesting that the information content of the conditioning signal can control this effect.

Feedback Distillation as on-policy knowledge transfer. When the feedback model is stronger than the student, Feedback Distillation acts as indirect knowledge distillation: the strong model provides information without needing to supply a full solution, unlike in standard distillation. As discussed in the introduction, the process is also inherently on-policy, as the training signal is computed on the student's generations, a property shown to improve generalization and reduce catastrophic forgetting compared to off-policy methods such as SFT (Shenfeld et al., 2026a). We provide in Appendix C.2 a comparison between using SFT and Feedback Distillation as an initialization for a GRPO run. Our results show that Feedback Distillation gets comparable results with an SFT-based pipeline while being on-policy. We believe this opens exciting research directions towards fully on-policy post-training of LLMs at scale. A further practical advantage over standard on-policy distillation (Agarwal et al., 2024) is that knowledge is transferred through natural-language feedback rather than logit matching, removing the need to host the feedback model or share its tokenizer.

6 Conclusion

We view Feedback Distillation as a promising mechanism that addresses fundamental limitations of existing approaches by injecting external knowledge, maintaining diversity, and providing fine-grained credit assignment. More broadly, current LLMs appear to have sufficient reasoning capabilities to analyze their own successes and failures in grounded environments like formal mathematics, and Feedback Distillation seems a promising direction to internalize this ability into the model's weights. Our work opens several directions for future work. Training stability remains a challenge for Feedback Distillation; developing more robust stabilization mechanisms is a natural next step toward scaling to longer training runs. Additionally, while our results in Lean theorem proving are encouraging, extending Feedback Distillation to other formal and informal reasoning tasks could further demonstrate the generality of the approach.

References

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

Anthropic. Introducing Claude opus 4.6, February 2026. URL `https://www.anthropic.com/news/claude-opus-4-6`. Accessed: 2026-04-14.

Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding

predictive architecture. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15619–15629, 2023.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, et al. Graph of thoughts: Solving elaborate problems with large language models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(16):17682–17690, 2024.

Xingyu Dang, Christina Baek, J Zico Kolter, and Aditi Raghunathan. Assessing diversity collapse in reasoning. In *Scaling Self-Improving Foundation Models without Human Supervision*, 2025.

Leonardo de Moura and Sebastian Ullrich. The Lean 4 theorem prover and programming language. In *Automated Deduction – CADE 28: 28th International Conference on Automated Deduction, Virtual Event, July 12–15, 2021, Proceedings*, page 625–635, Berlin, 2021. Springer.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent: A new approach to self-supervised learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 21271–21284. Curran Associates, Inc., 2020.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9729–9738, 2020.

Thomas Hubert, Rishi Mehta, Laurent Sartran, Miklós Z Horváth, Goran Žužić, Eric Wieser, Aja Huang, Julian Schrittwieser, Yannick Schroecker, Hussain Masoom, et al. Olympiad-level formal mathematical reasoning with reinforcement learning. *Nature*, pages 1–3, 2025.

Jonas Hübotter, Frederike Lübeck, Lejs Behric, Anton Baumann, Marco Bagatella, Daniel Marta, Ido Hakimi, Idan Shenfeld, Thomas Kleine Buening, Carlos Guestrin, and Andreas Krause. Reinforcement learning via self-distillation, 2026.

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

Jeonghye Kim, Xufang Luo, Minbeom Kim, Sangmook Lee, Dongsheng Li, and Yuqing Yang. Understanding reasoning in llms through strategic information allocation under uncertainty. *arXiv preprint arXiv:2603.15500*, 2026b.

Komal Kumar, Tajamul Ashraf, Omkar Thawakar, Rao Muhammad Anwer, Hisham Cholakkal, Mubarak Shah, Ming-Hsuan Yang, Phillip HS Torr, Fahad Shahbaz Khan, and Salman Khan. LLM post-training: A deep dive into reasoning large language models. *arXiv preprint arXiv:2502.21321*, 2025.

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

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. *arXiv preprint arXiv:2305.20050*, 2023.

Yong Lin, Shange Tang, Bohan Lyu, Ziran Yang, Jui-Hui Chung, Haoyu Zhao, Lai Jiang, Yihan Geng, Jiawei Ge, Jingruo Sun, Jiayun Wu, Jiri Gesi, Ximing Lu, David Acuna, Kaiyu Yang, Hongzhou Lin, Yejin Choi, Danqi Chen, Sanjeev Arora, and Chi Jin. Goedel-prover-V2: Scaling formal theorem proving with scaffolded data synthesis and self-correction, 2025.

Junqi Liu, Zihao Zhou, Zekai Zhu, Marco Dos Santos, Weikun He, Jiawei Liu, Ran Wang, Yunzhou Xie, Junqiao Zhao, Qiufeng Wang, et al. Numina-lean-agent: An open and general agentic reasoning system for formal mathematics. *arXiv preprint arXiv:2601.14027*, 2026.

Antoine Peyronnet, Fabian Gloeckle, and Amaury Hayat. Lemmabench: A live, research-level benchmark to evaluate LLM capabilities in mathematics. *arXiv preprint arXiv:2602.24173*, 2026.

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

Z. Z. Ren, Zhihong Shao, Junxiao Song, Huajian Xin, Haocheng Wang, Wanjia Zhao, Liyue Zhang, Zhe Fu, Qihao Zhu, Dejian Yang, Z. F. Wu, Zhibin Gou, Shirong Ma, Hongxuan Tang, Yuxuan Liu, Wenjun Gao, Daya Guo, and Chong Ruan. Deepseek-prover-V2: Advancing formal mathematical reasoning via reinforcement learning for subgoal decomposition, 2025.

Stephane Ross and Drew Bagnell. Efficient reductions for imitation learning. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9, pages 661–668, 2010.

Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Romain Sauvestre, Tal Remez, Jérémy Rapin, et al. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*, 2023.

Marco Dos Santos, Haiming Wang, Hugues de Saxcé, Ran Wang, Mantas Baksys, Mert Unsal, Junqi Liu, Zhengying Liu, and Jia Li. Kimina lean server: Technical report, 2025. URL `https://arxiv.org/abs/2504.21230`.

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

Idan Shenfeld, Mehul Damani, Jonas Hübotter, and Pulkit Agrawal. Self-distillation enables continual learning, 2026a.

Idan Shenfeld, Jyothish Pari, and Pulkit Agrawal. Rl's razor: Why online reinforcement learning forgets less. In *International Conference on Learning Representations*, volume 2026, pages 59839–59864, 2026b.

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

Qwen Team. Qwen3 technical report, 2025.

Qwen Team. Qwen3.5: Accelerating productivity with native multimodal agents, February 2026. URL `https://qwen.ai/blog?id=qwen3.5`.

The mathlib Community. The Lean Mathematical Library. In *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs*, CPP 2020, New Orleans, 2020. ACM.

Trieu H. Trinh, Yuhuai Wu, Quoc V. Le, He He, and Thang Luong. Solving olympiad geometry without human demonstrations. *Nature*, 625(7995):476–482, 2024.

George Tsoukalas, Jasper Lee, John Jennings, Jimmy Xin, Michelle Ding, Michael Jennings, Amitayush Thakur, and Swarat Chaudhuri. PutnamBench: Evaluating neural theorem-provers on the putnam mathematical competition. In *Advances in Neural Information Processing Systems*, volume 37, pages 11545–11569. Curran Associates, Inc., 2024.

Sumanth Varambally, Thomas Voice, Yanchao Sun, Zhifeng Chen, Rose Yu, and Ke Ye. Hilbert: Recursively building formal proofs with informal reasoning, 2026.

Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and Quentin Gallouédec. TRL: Transformers Reinforcement Learning, 2020. URL `https://github.com/huggingface/trl`.

Fang Wu, Weihao Xuan, Ximing Lu, Mingjie Liu, Yi Dong, Zaid Harchaoui, and Yejin Choi. The invisible leash: Why RLVR may or may not escape its origin, 2026.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023.

Huaiyuan Ying, Zijian Wu, Yihan Geng, Zheng Yuan, Dahua Lin, and Kai Chen. Lean workbook: A large-scale Lean problem set formalized from natural language math problems, 2025.

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in LLMs beyond the base model?, 2025.

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

Kunhao Zheng, Jesse Michael Han, and Stanislas Polu. MiniF2F: A cross-system benchmark for formal Olympiad-level mathematics, 2022.

Acknowledgments and Disclosure of Funding

We are grateful to Julia Kempe for her help and valuable advice throughout the writing of this article.

A Setup and Implementation

A.1 Training hyperparameters

Table 1 lists the hyperparameters used in all Feedback Distillation and GRPO experiments reported in the paper.

Table 1: Training hyperparameters for Feedback Distillation and GRPO experiments.

Hyperparameter	Feedback Distillation	GRPO
Learning rate	1×10^{-5}	1×10^{-7}
Prompts per batch	128	128
Samples per prompt	1	5
Effective batch size	128	640
EMA coefficient α	0.9	—
Top-K truncation	25	—
KL regularization coeff.	—	0.001
Max response length	8192	8192
Max prompt length	2048	2048
Linear learning rate warmup	50 steps	50 steps

A.2 Tool use ablation

Figure 7 compares training with and without tool access for both Claude Feedback Distillation and GRPO with Qwen3-8B. In both cases, tool use leads to higher MiniF2F accuracy and better training accuracy, confirming that iterative interaction with the Lean compiler is a driver of performance (Liu et al., 2026).

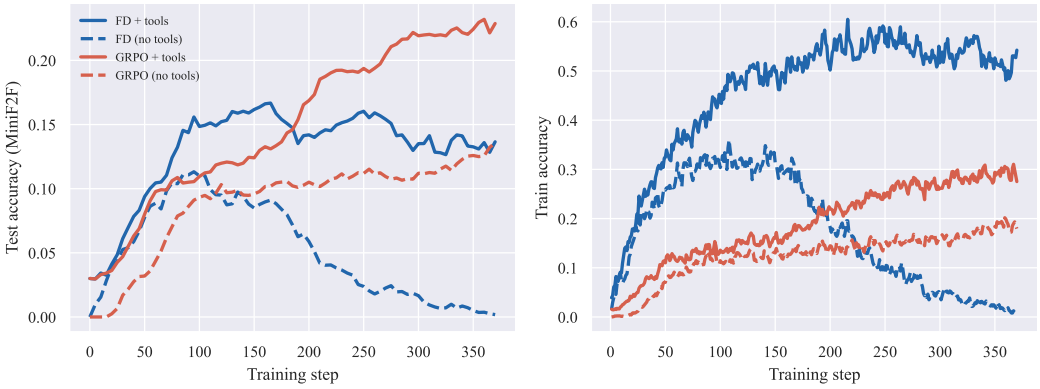

Figure 7: Qwen3-8B MiniF2F accuracy and training reward for Feedback Distillation and GRPO, with and without tool access. Tool use improves performance for both methods.

A.3 Feedback prompt

The following prompt is given to the feedback model (Claude Opus 4.6) along with the formal statement x, the model's attempt y (including tool calls and their outputs), and the final Lean compiler output.

> You are an expert feedback tutor. Given a problem and a student's attempt, write concise, actionable pieces of advice that will provide general feedback from the student's experience. Write between 1 and 3 numbered pieces of feedback about errors, things that were useful, or things that were not. The feedback must be self-contained and useful to someone who does not have access to the student's answer. It must be framed as orders, like "Do X" or "Don't do X", without mentioning the student. Each piece of feedback must be between 10

and 50 words. The goal is not to comment on the performance of the student, but to provide useful feedback for someone who might want to try the same problem in the future. Do not use "you", "your", or the second person in general. It must be useful to someone who does not know there is a student. It must encourage the usage of tools. Do not output anything other than the feedback — no introduction, directly the 1. of the first piece of advice. If the answer seems truncated at the end, it may be because the student reached the token limit; in this case, order to be more concise.

A.4 Compute comparison between GRPO and Feedback Distillation

Table 2 reports the compute spent by Claude Feedback Distillation and by GRPO to reach step 200. Feedback Distillation draws a single sample per prompt where GRPO draws five, so it reaches the same step count having generated $5\times$ fewer trajectories in $1.8\times$ less wall-clock time. At current prices this is roughly \$250 of GPU compute against \$440 for GRPO, but Feedback Distillation additionally queries an external feedback model, at approximately \$1,100, so its saving in compute is more than offset by the API cost. We give these figures as a point of information only: they track API pricing, which keeps falling as cheaper frontier models of comparable quality become available.

Table 2: Training cost of Claude Feedback Distillation and GRPO, accumulated over the first 200 training steps. Feedback Distillation reaches step 200 with $5\times$ fewer rollouts and $1.8\times$ less wall-clock.

Metric (cumulative through step 200)	Claude FD	GRPO	GRPO / FD
Rollouts per training step	128	640	$5.0\times$
Total rollouts	25,600	128,000	$5.0\times$
Wall-clock (h)	15.85	27.74	$1.8\times$
Generation (h)	5.15	11.39	$2.2\times$
Feedback (h)	1.39	0	—
Generated response tokens (M)	137.1	801.0	$5.8\times$
Lean tool calls (k)	423.6	2,064.5	$4.9\times$
Mean tool calls per trajectory	16.5	16.1	$1.0\times$
Feedback-model calls	25,600	0	—
Feedback input tokens (M)	≈ 168.2	0	—
Feedback output tokens (M)	≈ 7.68	0	—

B Loss derivation

Since gradients are not propagated through the sampling of y (stop-gradient), the gradient of $\mathcal{L}_\theta$ with respect to θ reduces to differentiating only through π_θ inside the KL:

$$\nabla_\theta \mathcal{L}_\theta \;=\; -\mathbb{E}_{x\sim D,\, y\sim\pi_\theta(\cdot|x)} \left[\sum_{t=1}^{|y|} \sum_{a\in V} \pi'_{\mu,y}(a \mid x, y_{<t})\; \nabla_\theta \log \pi_\theta(a \mid x, y_{<t}) \right].$$

Since the entropy of the teacher $H(\pi'_{\mu,y})$ does not depend on θ, the KL divergence and the cross-entropy have the same gradient with respect to θ. Expanding $\pi'_{\mu,y}$ by its definition, we obtain the gradient-equivalent loss used in practice:

$$\hat{\mathcal{L}}_\theta \;=\; -\sum_{t=1}^{|y|} \sum_{a\in V} \pi_\mu(a \mid x,\, F(y),\, y_{<t})\; \log \pi_\theta(a \mid x,\, y_{<t}).$$

C Additional Results and Analysis

C.1 EMA sweep for Qwen3.5-9B

Consistent with the observations in Figure 5 for Qwen3-8B, we find in Figure 8 that the same trend holds for Qwen3.5-9B: more aggressive EMA updates lead to faster early learning, but induce greater instability and ultimately limit peak performance.

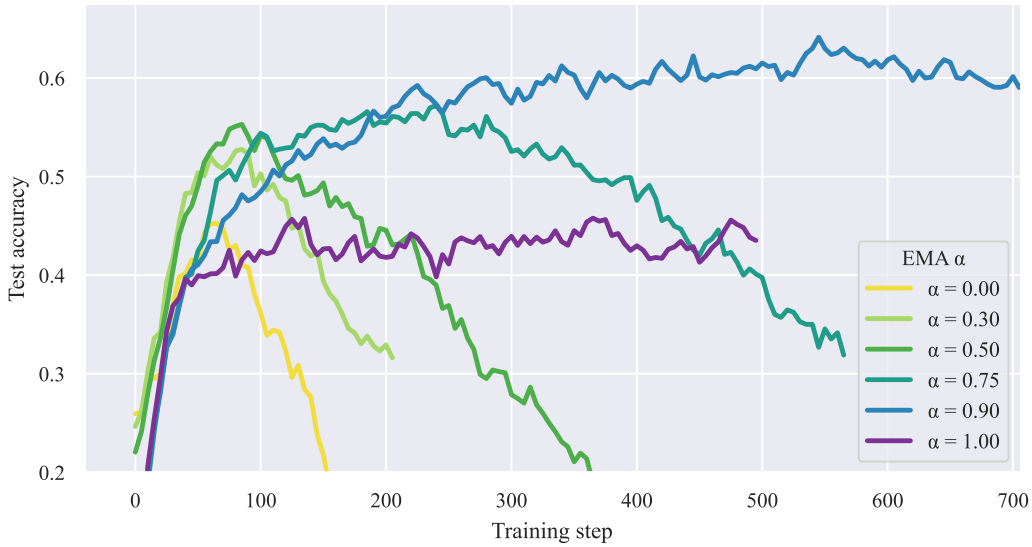

Figure 8: LeanWorkbook test accuracy for Qwen3.5-9B during training for different values of the EMA interpolation parameter α. A more aggressive EMA (i.e., a lower value of α) leads to faster training but also more instability. $\alpha = 1$ is the same as freezing the teacher weights.

C.2 Comparison between Feedback Distillation and SFT

The GRPO over SFT curve in Figure 9 starts from a supervised-finetuned Qwen3.5-9B rather than from a Feedback Distillation checkpoint, so that the two initializations can be compared under the same GRPO recipe.

The SFT data was made by prompting Claude to prove Lean Workbook theorems with access to Lean tools, recording both its reasoning and its final proof. 9,494 training examples are thus built, with a median of roughly 1,900 characters of reasoning and 170 characters of proof.

Training used TRL's `SFTTrainer` (von Werra et al., 2020) on a single 8×H200 node with DeepSpeed ZeRO-2; Table 3 lists the hyperparameters. At an effective batch size of 64, one epoch is 148 optimizer steps, so the checkpoint at step 200 we branch from corresponds to 1.35 epochs, or 12,800 examples seen.

The goal here is to compare Feedback Distillation with SFT as a first training phase before a GRPO training. Both methods get similar results, and are a way to distill Claude Lean knowledge, either off-policy by training on its generated proofs, or on-policy by using its feedback only (and not whole solution) as privileged information in Feedback Distillation. We see Feedback Distillation as a different and reasonable way to distill knowledge in an on-policy way.

C.3 Feedback formulation

We investigate how the formulation of feedback affects training performance. To this end, we evaluate four different prompt styles for Claude Opus 4.6 as the feedback model, while keeping all other hyperparameters fixed. The prompts are ordered from (1) to (4) by increasing level of structure and guidance, with each successive prompt introducing additional instructions: (1) a minimal prompt requesting feedback, (2) the addition of formatting constraints (e.g., length and number of feedback

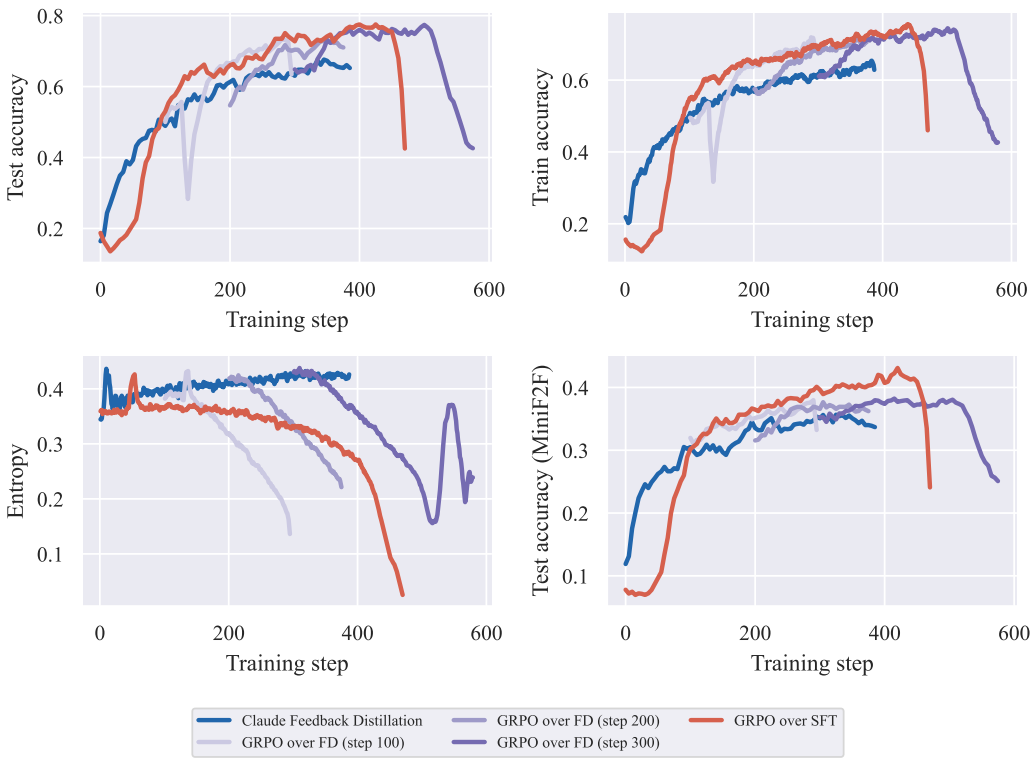

Figure 9: **GRPO over SFT versus Feedback Distillation on Qwen3.5-9B.** LeanWorkbook test accuracy, train accuracy, policy entropy and MiniF2F test accuracy, for Feedback Distillation, GRPO resumed from Feedback Distillation checkpoints at 100, 200 and 300 steps, and GRPO resumed from a SFT checkpoint.

Table 3: Hyperparameters for the supervised finetuning that initializes the GRPO over SFT run.

Hyperparameter	Value
Base model	Qwen3.5-9B
Learning rate	1×10^{-5}
Schedule	cosine, warmup ratio 0.03
Optimizer	AdamW, $\beta = (0.9, 0.999)$, weight decay 0
Gradient clipping	1.0
Effective batch size	64 (2×4 accumulation $\times$ 8 GPUs)
Epochs	1.35
Max sequence length	8192

items), (3) a requirement that the feedback be self-contained and useful to a reader without access to the student's answer, and (4) additional guidance encouraging the use of tools and limiting generation length in case of truncation. We use Prompt (4) in all experiments.

Prompt templates. We provide below the four prompt formulations used in this study:

- **Prompt (1): Minimal feedback request**

  You are an expert feedback tutor. Given a problem and a student's attempt, write concise, actionable pieces of advice based on what the student did.

- **Prompt (2): + Formatting constraints**

  You are an expert feedback tutor. Given a problem and a student's attempt, write concise, actionable pieces of advice based on what the student did. Write between 1 and 3 numbered

feedbacks about errors, things that were useful, or things that were not. They must not be too long — between 10 and 50 words each. Do not output anything else than the feedbacks, no introduction, directly the "1." of the first advice.

- **Prompt (3): + Self-contained feedback requirement**

  You are an expert feedback tutor. Given a problem and a student's attempt, write concise, actionable pieces of advice based on what the student did. Write between 1 and 3 numbered feedbacks about errors, things that were useful, or things that were not. They must not be too long — between 10 and 50 words each. Do not output anything else than the feedbacks, no introduction, directly the "1." of the first advice. The feedback must be self-contained and useful to someone who does not have access to the student's answer. Don't use "you", "your", or the second person in general — it must be useful to someone who doesn't know there is a student. It must be framed as orders, like "Do x" or "Don't do x", without mentioning the student. The goal is not to comment on the performance of the student, but to provide useful guidance for someone who might want to try the same problem in the future.

- **Prompt (4): + Tool use and truncation guidance**

  You are an expert feedback tutor. Given a problem and a student's attempt, write concise, actionable pieces of advice that will provide general feedback from the student's experience. Write between 1 and 3 numbered pieces of feedback about errors, things that were useful, or things that were not. The feedback must be self-contained and useful to someone who does not have access to the student's answer. It must be framed as orders, like "Do X" or "Don't do X", without mentioning the student. Each piece of feedback must be between 10 and 50 words. The goal is not to comment on the performance of the student, but to provide useful feedback for someone who might want to try the same problem in the future. Do not use "you", "your", or the second person in general. It must be useful to someone who does not know there is a student. It must encourage the usage of tools. Do not output anything other than the feedback — no introduction, directly the 1. of the first piece of advice. If the answer seems truncated at the end, it may be because the student reached the token limit; in this case, order to be more concise.

Figure 10 shows that the formulation of the feedback has a significant impact on training dynamics. More structured prompts, which provide richer and more constrained feedback, tend to yield slower initial performance gains but result in more stable training and higher peak performance. In particular, prompts that encourage generic, self-contained feedback (Prompts 3 and 4) appear to be especially beneficial.

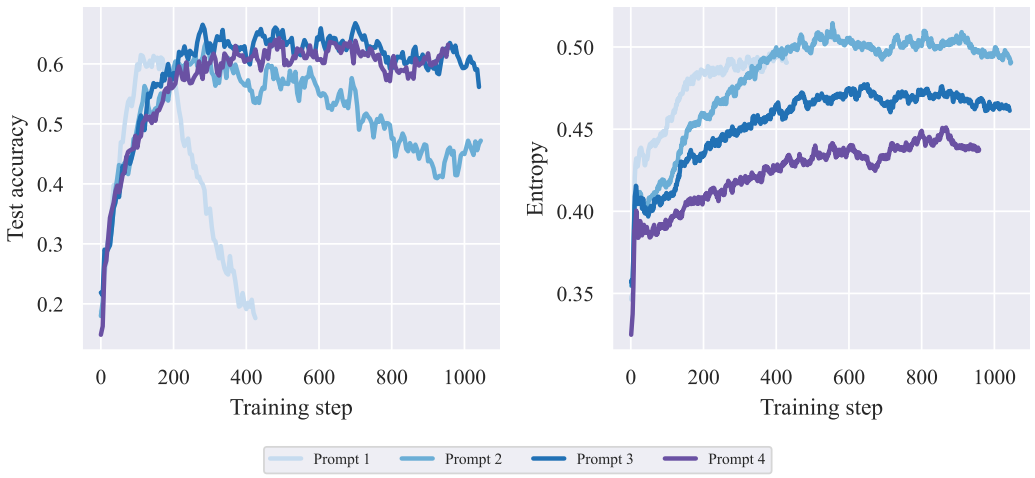

Figure 10: Impact of feedback prompt formulation on training performance. Increasing the level of structure in the prompt improves stability and peak performance.

C.4 Epistemic verbalization and response length

Following Kim et al. (2026b), we measure epistemic verbalization by counting the average number of uncertainty words generated by the student during training. Figure 11 reports the per-step occurrence counts of three representative epistemic word categories (see Section 5.3) across training runs. When the feedback consists of a ground-truth solution, all three counts drop sharply early in training, reflecting the suppression of epistemic verbalization. By contrast, when feedback is framed as a critique by a language model, whether the model itself or a stronger one, the counts do not follow the same dynamic throughout training (especially when using Claude as the feedback model), indicating that it is possible to preserve the epistemic ability of the student. We attribute this to the student mimicking the teacher's style, which is itself shaped by the feedback: a directive signal such as a worked-out solution leads the teacher to reason with greater confidence, while a critique preserves more uncertainty expression.

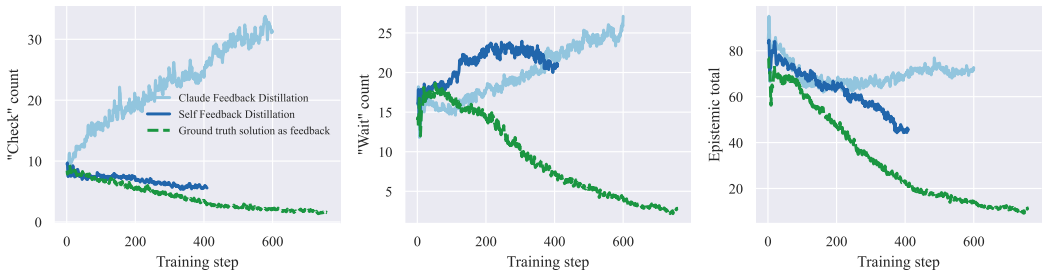

Figure 11: Occurrence counts of epistemic words (*"check"*, *"wait"*, and a combined epistemic total aggregating *wait, hmm, perhaps, maybe, actually, alternatively, seems, might, likely, check*) over training steps for Qwen3-8B. Ground-truth feedback causes a marked decline in all three counts, whereas feedback framed as a critique by a language model — regardless of whether it comes from the student itself or a stronger model — preserves the student's epistemic verbalization.

Critique-based feedback is also associated with longer responses than ground-truth feedback (Figure 12), consistent with models that maintain diversity in their generated trajectories rather than converging to short, confident proofs.

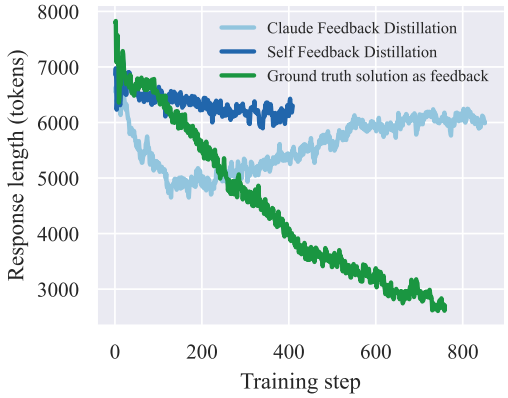

Figure 12: Qwen3-8B mean response length (in tokens) over training steps. Critique-based feedback (Claude and self-feedback) is associated with longer responses than ground-truth feedback, reflecting preserved diversity in generated trajectories.

D Feedback examples

The following are four representative feedback messages produced by Claude Opus 4.6 at step 100 of Qwen3-8B training with Feedback Distillation. Each box corresponds to a different theorem-proving attempt.

1. When proving inequalities involving absolute values in Lean 4, use `rcases le_or_lt b a` and `rcases le_or_lt c a` to split into cases based on the sign of each expression inside the absolute values, then simplify using `abs_of_nonneg` and `abs_of_nonpos` before finishing with `linarith`.
2. Always write and test intermediate Lean files using the `lean_write_file` tool rather than reasoning entirely in one's head — this ensures syntax errors and tactic failures are caught early and iteratively corrected.
3. Structure the proof by first splitting on `le_or_lt (b + c) a` to handle `|a - b - c|`, then within the `a < b + c` branch, further split on the signs of `a - b` and `a - c` to reduce all absolute values to linear expressions before applying `linarith`.

1. When working with natural number division in Lean 4, avoid assuming `field_simp` will clear denominators as it does for fields; instead, use `Nat.div_mul_cancel` or establish divisibility (`n ∣ 175`) first, then rewrite to eliminate the division before algebraic manipulation.
2. Use `omega` or `interval_cases` after bounding n (e.g., from the equation deduce $n \leq 175$) to let Lean automatically check all possible natural number values, which is often simpler than manual algebraic factoring for small search spaces.
3. Write candidate proofs to a file using `lean_write_file` and check them with `lean_check_file` iteratively, fixing errors based on compiler output rather than reasoning abstractly about tactic behavior.

1. Avoid division in Lean proofs over reals; instead, multiply through to keep all expressions polynomial, enabling `ring` and `nlinarith` to close goals without needing to prove denominators are nonzero.
2. Derive the key intermediate fact `x * y + z * (x + y) = 0` by showing `(x - y) * (x * y + z * (x + y)) = 0` via `have : ... = x^2 * (y + z) - y^2 * (z + x) := by ring` combined with `linarith [h1, h2]`, then use $\texttt{mul_left_cancel}_0$ with `sub_ne_zero.mpr hxy`.
3. After establishing `x * y + z * (x + y) = 0`, close the final goal using `nlinarith [h1, h2, sq_nonneg x, sq_nonneg y, sq_nonneg z, sq_nonneg (x - y)]` to let the nonlinear arithmetic solver combine all hypotheses without manual algebraic manipulation.

1. Decompose the inequality into proving $a^2 + b^2 + c^2 + d^2 \geq 4$ (AM-GM on squares with product $(abcd)^2 = 1$) and $ab + ac + ad + bc + bd + cd \geq 6$ (AM-GM on six pairwise products with product $(abcd)^3 = 1$), then combine with `linarith`.
2. When using `nlinarith`, supply auxiliary lemmas as hints — e.g., `sq_nonneg (a*b - 1)`, `sq_nonneg (a*c - 1)`, `mul_pos ha hb`, and the hypothesis `h` — so the solver can close the nonlinear arithmetic gap created by the constraint $abcd = 1$.
3. Always write a complete, syntactically valid Lean file and verify it compiles using the tool before submitting; an incomplete or unreadable file scores zero regardless of mathematical correctness.

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 claims made match experimental results, and reflect how much the results can be expected to generalize to other settings.

   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: Training instability is discussed in Section 5.2, and broader limitations and future directions are addressed in Section 6.

   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 does not include theoretical results. The loss derivation in Appendix B is a straightforward algebraic identity.

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: All implementation details of the algorithm are provided, as well as hyperparameters (see Section 3 and Appendix A.1).

   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: [No]

Justification: All datasets and models are publicly available. The code is not made available at submission time but will be released with the camera-ready version.

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 main training and evaluation details are provided in Section 4.2 and Appendix A.1. Remaining details (e.g., optimizer choice) will be provided with the code release.

   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: [No]

   Justification: Due to the high computational cost of each training run (GPU compute and API calls for feedback generation), we do not report error bars. We mitigate this by validating our main conclusions on two base models (Qwen3-8B and Qwen3.5-9B).

   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: This information is available in Subsection 4.2.

   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 research conforms to the NeurIPS Code of Ethics.

   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: [N/A]

    Justification: This work is foundational research on training methods for formal theorem proving, with no direct negative societal impact.

    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 models or datasets with high risk for misuse.

    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: All models, datasets and code used in the project are properly credited.

    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 new datasets, models, or other assets.

    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 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 subjects research.

    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 use of Claude Opus 4.6 as the feedback model is described in Sections 3 and 4.2.

    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.