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 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, because it trains on off-policy demonstrations, it can cause catastrophic forgetting outside the narrow fine-tuning distribution (see, e.g., Ross and Bagnell, 2010; Agarwal et al., 2024; Shenfeld et al., 2026). Moreover, contrary to the popular belief that RL training leads to new reasoning capabilities, 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 essential for success. Finally, GRPO provides only trajectory-level rewards, which offer no learning signal when all samples fail.

Preprint.

Statement to prove

```
theorem mathd_algebra_480 (f : ℝ → ℝ)
    (h₀ : ∀ x < 0, f x = -x^2 - 1)
    (h₁ : ∀ x, 0 ≤ x ∧ x < 4 → f x = 2)
    (h₂ : ∀ x ≥ 4, f x = Real.sqrt x) : f Real.pi = 2 := by
```

Student generation without feedback (truncated)

```
(...) Lean, the tactics might involve using the lemmas. Let me try to write the proof.
The code would be:
by
have h_pi_nonneg : 0 ≤ π := Real.pi_nonneg
have h_pi_lt_4 : π < 4 := Real.pi_lt_4
have h_pi_in_range (...)
```

Claude's feedback (seen by the teacher model during distillation)

> Use `Real.pi` instead of the unicode π symbol in Lean proofs — Lean's Mathlib lemmas (e.g., `Real.pi_nonneg`, `Real.pi_lt_four`) refer to `Real.pi`, so using π directly may cause unification failures.

Same student generation colored by $\log p_{\text{teacher}} - \log p_{\text{student}}$

```
(...) Lean , the tactics might involve using the le mmas . Let me try to write the proof .
The code would be :
by
have h _pi _non neg : 0 ≤ π := Real .pi _non neg
have h _pi _lt _4 : π < 4 := Real .pi _lt _4
have h _pi _in _range (...)
```

Probability comparison

For the token π, the student gives the unicode symbol substantial probability: **56.1%**. By contrast, the teacher strongly prefers Lean's canonical form, assigning **98.1%** to `Real`, versus only **1.8%** to π and **0.1%** to ℝ.

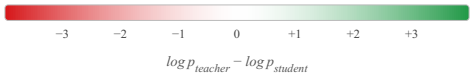

Figure 1: Example of Feedback Distillation on a Lean statement from the MiniF2F dataset (Zheng et al., 2022), with Qwen3-8B as the student. Claude provides feedback on the student's attempt. The student's answer is then colored token-by-token by $\log p_{\text{teacher}} - \log p_{\text{student}}$: green tokens are preferred by the teacher, red tokens by the student, and white tokens are assigned roughly equal probability. 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.

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., 2026; Zhao et al., 2026), which trains the model to mimic its own outputs when conditioned on privileged information, effectively distilling the prompting strategy directly into the model's weights. This on-policy approach is attractive because it provides fine-grained credit assignment at the token level rather than distributing reward uniformly across entire trajectories (see example in Figure 1). Moreover, it can apply beyond verifiable reward settings since it does not directly rely on verifier signals, though in our setting we also leverage verifier outputs.

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 frozen copy of the model or a third-party LLM 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, Qwen3.5-9B achieves 59% pass@1 test accuracy on a test split of LeanWorkbook with GRPO alone compared to 75% 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. (2026); 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., 2026; 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 $x \sim D$, we define the teacher as

$$\pi'_{\mu,y}(\cdot \mid x) \;=\; \pi_\mu(\cdot \mid x,\, F(y)),$$

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 $\pi'_{\mu,y}$ 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:

$$\mathcal{L}_\theta \;=\; \mathbb{E}_{x\sim D,\; y\sim\pi_\theta(\cdot|x)}\Bigg[\sum_{t=1}^{|y|}\mathrm{KL}\big(\pi'_{\mu,y}(\cdot\mid x,y_{<t})\;\|\;\pi_\theta(\cdot\mid x,y_{<t})\big)\Bigg],$$

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

$$\mathrm{KL}\big(\pi'_{\mu,y}(\cdot\mid x,y_{<t})\;\|\;\pi_\theta(\cdot\mid x,y_{<t})\big) \;=\; \sum_{a\in V}\pi'_{\mu,y}(a\mid x,y_{<t})\log\frac{\pi'_{\mu,y}(a\mid x,y_{<t})}{\pi_\theta(a\mid x,y_{<t})}.$$

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 the main paper are given for Qwen3.5-9B and in the appendix for Qwen3-8B.

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. 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.

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

Figure 2 shows the training accuracy, test accuracy on LeanWorkbook and MiniF2F, and policy entropy over the course of training for both GRPO and Feedback Distillation. We also experiment with *GRPO over Feedback Distillation*, meaning that we take a checkpoint from Feedback Distillation and continue training with GRPO. Overall, **Feedback Distillation is able to learn efficiently while maintaining diversity in the generated trajectories**, as evidenced by the observations detailed below. This is 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 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.

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.

Number of used tactics. A concrete manifestation of Feedback Distillation's greater trajectory diversity is the rate at which the model uses new Lean tactics and 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 tactics and lemmas invoked, compared to GRPO.

Training instabilities. We observe training instabilities with GRPO and GRPO over Feedback Distillation. This behavior was already noted in prior works (Arnal et al., 2025; Simoni et al., 2025). We check that the maximum performance of GRPO over Feedback Distillation cannot be reproduced by GRPO alone simply by tuning the learning rate to delay instability (see Appendix C.1 for details). We also note that GRPO suffers from fewer instabilities with Qwen3-8B (see Appendix C.2). Importantly, the observation that GRPO combined with Feedback Distillation outperforms GRPO alone also holds for Qwen3-8B, confirming that the improvements brought by Feedback Distillation are not merely a byproduct of GRPO instabilities.

Sample efficiency. The results of Feedback Distillation in terms of sample efficiency are encouraging. In particular, when using Qwen3.5-9B as a base model, Feedback Distillation is significantly more sample-efficient in the early stages of training than GRPO, although GRPO eventually catches up before crashing. Note that the batch size used for GRPO is five times larger than that used for Feedback Distillation, due to the group size of 5 (see Appendix A.1). For Qwen3-8B (see Appendix C.2),

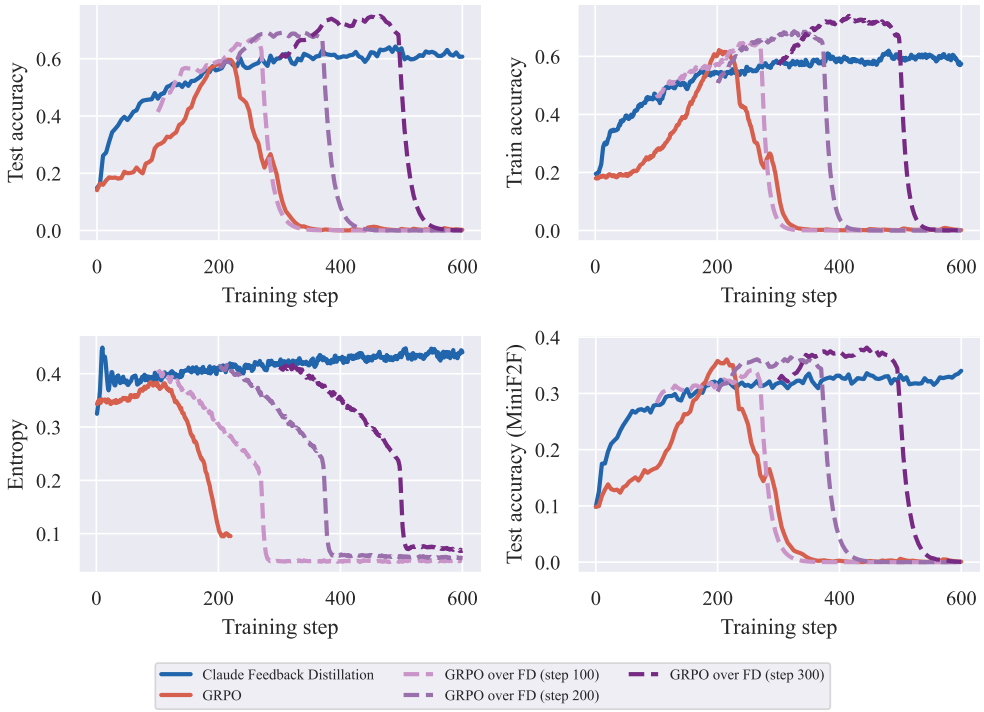

Figure 2: Test accuracy, train accuracy, policy entropy, and MiniF2F accuracy for Claude Feedback Distillation, GRPO, and GRPO initialized from three different checkpoints from Claude Feedback Distillation. GRPO over Feedback Distillation outperforms either method alone.

we observe the same behavior, with Feedback Distillation being more sample-efficient in the early stages of training. We leave a more thorough assessment of sample efficiency for future work.

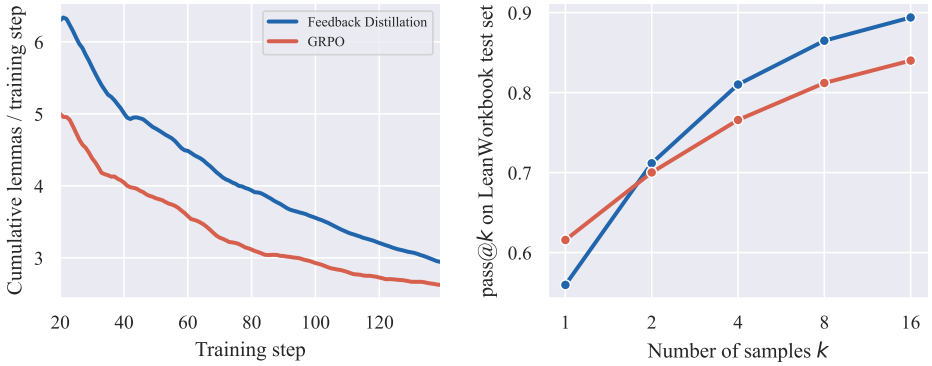

Figure 3: (Left) Rate of discovery of Mathlib lemmas: cumulative number of distinct lemmas used in correct proofs during training, divided by the training step. Feedback Distillation leads to faster discovery of new lemmas from the Mathlib library. (Right) Pass@k scaling for two checkpoints at 200 steps of training with GRPO and Feedback Distillation (from the experiment in Figure 2), selected at peak GRPO performance where both methods have similar pass@1 accuracy. Feedback Distillation scales better with k, confirming that its higher entropy translates into greater answer diversity.

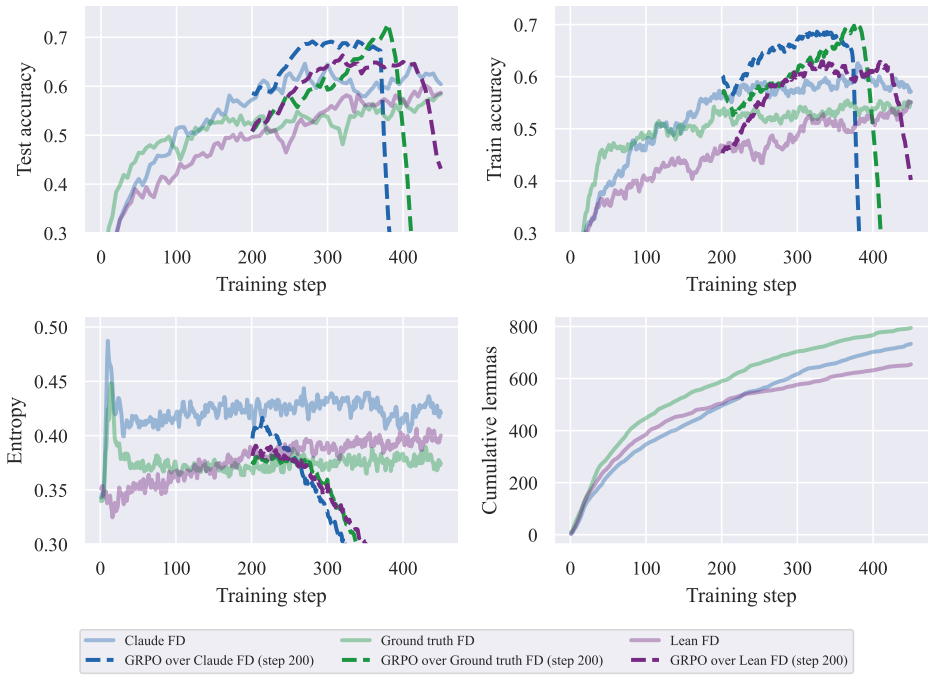

Figure 4: Feedback Distillation and GRPO over Feedback Distillation for different feedback sources: Claude, a ground truth Lean solution, and Lean compiler output. Claude and ground truth feedback sources are a better initialization for the GRPO training.

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

We consider two baselines on top of Claude for the feedback source: ground truth feedback (Shenfeld et al., 2026; Zhao et al., 2026) and Lean output (Hübotter et al., 2026). 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. Notably, using Claude as a feedback model induces higher entropy than relying on ground-truth solutions, which we attribute to the use of hints and corrections rather than fully specified answers. We discuss the impact of prompt formulation in Section 5.3 and leave further prompt engineering to future work.

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 enables a fair comparison with GRPO, as it does not introduce any additional source of information, and thus isolates the model's ability

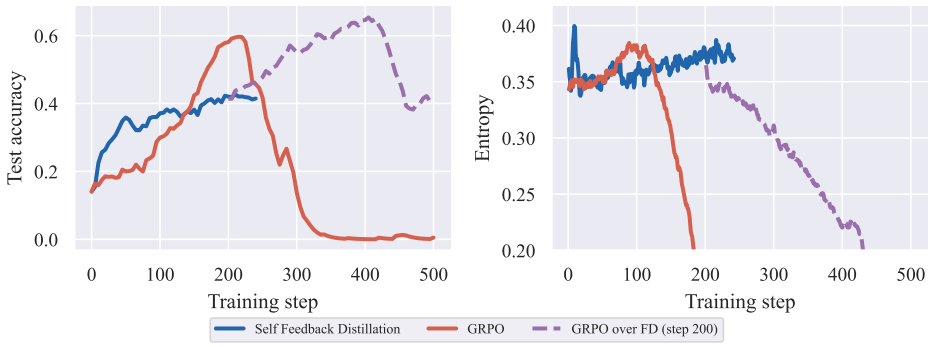

Figure 5: GRPO over Self-Feedback Distillation: when the feedback model is the same as the one we train (Qwen3.5-9B), GRPO over Self-Feedback Distillation still outperforms GRPO alone.

to improve based on its own analysis of its generations. As shown in Figure 5, combining GRPO with Self-Feedback Distillation leads to higher peak performance than GRPO alone, suggesting that the model is able to extract useful learning signals from its own feedback that are not discovered by GRPO in isolation. However, performance remains below that obtained with Lean compilation feedback, despite the feedback model having access to this information in its context. This suggests that further improvements may be achieved by refining the content and formulation of the feedback, pointing to promising directions for future work.

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 6 a sweep over α. We find that more aggressive updates lead to faster learning early in training, while $\alpha = 1$ avoids instability at the cost of slow learning. With our training budget, $\alpha = 0.9$ provides the best tradeoff. 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 7, 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.

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

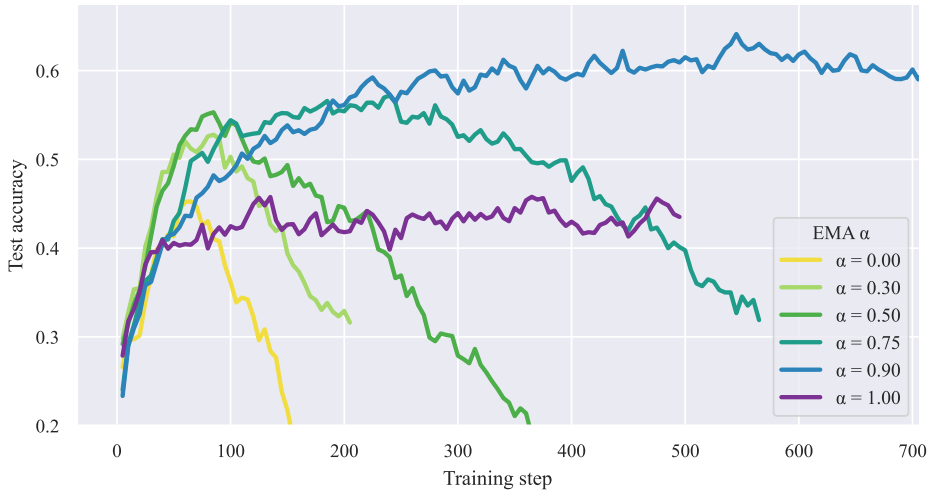

Figure 6: Test accuracy for different values of the EMA interpolation parameter α. A more aggressive EMA (i.e., a lower value of α) leads to faster training but also more instability. $\alpha = 1$ is the same as freezing the teacher weights. We use $\alpha = 0.9$ for all the experiments.

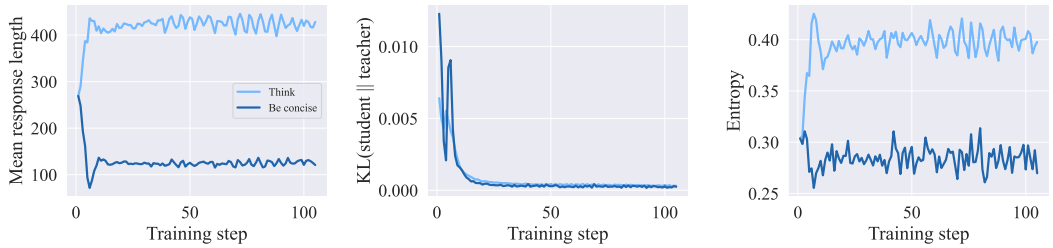

Figure 7: Effect of constant feedback on response length and policy entropy. "Think" feedback increases response length while "Be concise" decreases it. The KL divergence between student and teacher converges to zero, confirming that the feedback is internalized into the model's weights.

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. 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., 2026). 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

While our current results do not yet match state-of-the-art SFT+GRPO pipelines trained at scale, 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 offers a way to internalize this self-reflective 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.

Charles Arnal, Gaetan Narozniak, Vivien Cabannes, Yunhao Tang, Julia Kempe, and Rémi Munos. Asymmetric reinforce for off-policy reinforcement learning: Balancing positive and negative rewards. In *Advances in Neural Information Processing Systems*, 2025.

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, 2026.

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.

Marco Simoni, Aleksandar Fontana, Giulio Rossolini, Andrea Saracino, and Paolo Mori. Gtpo: Stabilizing group relative policy optimization via gradient and entropy control. *arXiv preprint arXiv:2508.03772*, 2025.

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.

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 8 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).

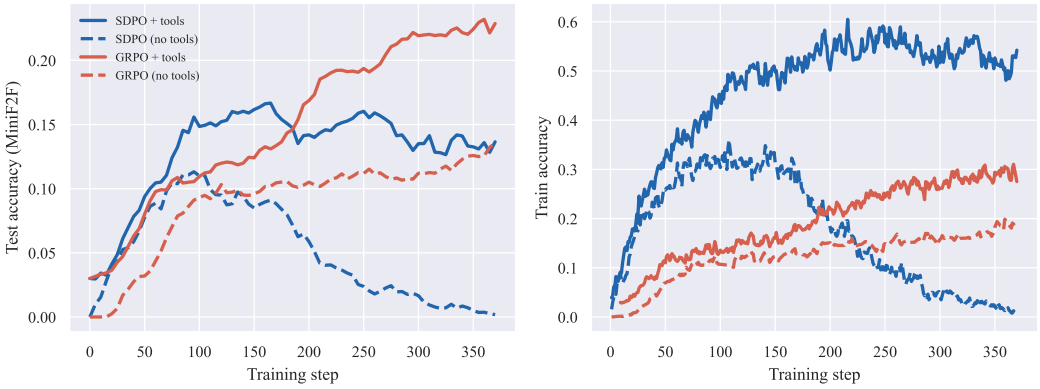

Figure 8: Qwen3-8B MiniF2F accuracy and training reward for Feedback Distillation and GRPO, with and without tool access. Tool use improves performance for both methods.

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.

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 Learning rate sweep for GRPO

In this section, we verify that the best performance achieved by GRPO is consistent with the results shown in Figure 2, regardless of the choice of learning rate around 10^{-7}. To this end, we evaluate two additional values, $3 \cdot 10^{-7}$ and $3 \cdot 10^{-8}$. As shown in Figure 9, all three learning rates lead to similar peak performance.

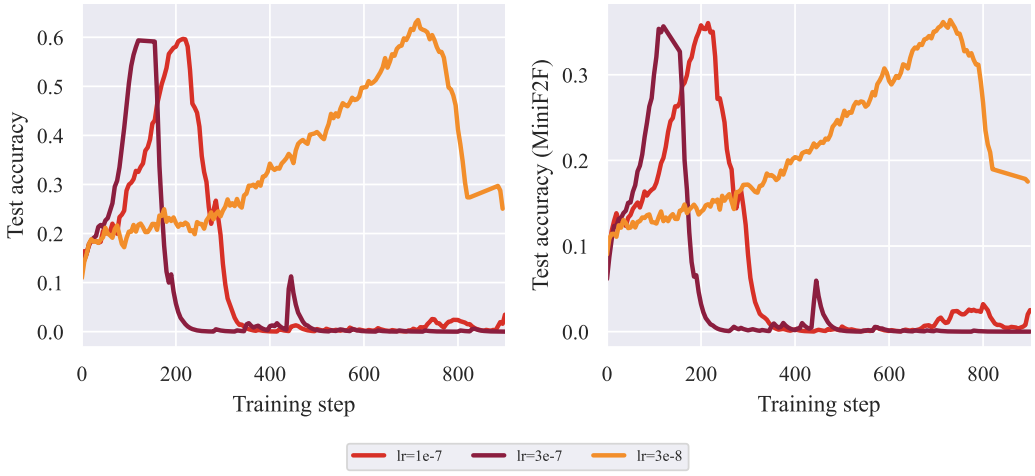

Figure 9: Accuracy on the LeanWorkbook test set and MiniF2F when training with GRPO using three different learning rates. All configurations reach similar peak performance.

C.2 Results with Qwen3-8B

We confirm in this section that the same conclusions hold for Qwen3-8B as for Qwen3.5-9B.

GRPO over Feedback Distillation. Training is more stable with Qwen3-8B, and we still observe the same conclusion as in Figure 2: GRPO over Feedback Distillation outperforms GRPO alone (we also use Claude Opus 4.6 as the feedback model for Feedback Distillation).

Figure 10: MiniF2F accuracy and policy entropy for Claude Feedback Distillation, GRPO, and GRPO initialized from a checkpoint from Claude Feedback Distillation, for Qwen3-8B. GRPO over Feedback Distillation outperforms either method alone.

EMA and training instability. Consistent with the observations in Figure 6 for Qwen3.5-9B, we find in Figure 11 that the same trend holds for Qwen3-8B: more aggressive EMA updates lead to faster early learning, but induce greater instability and ultimately limit peak performance.

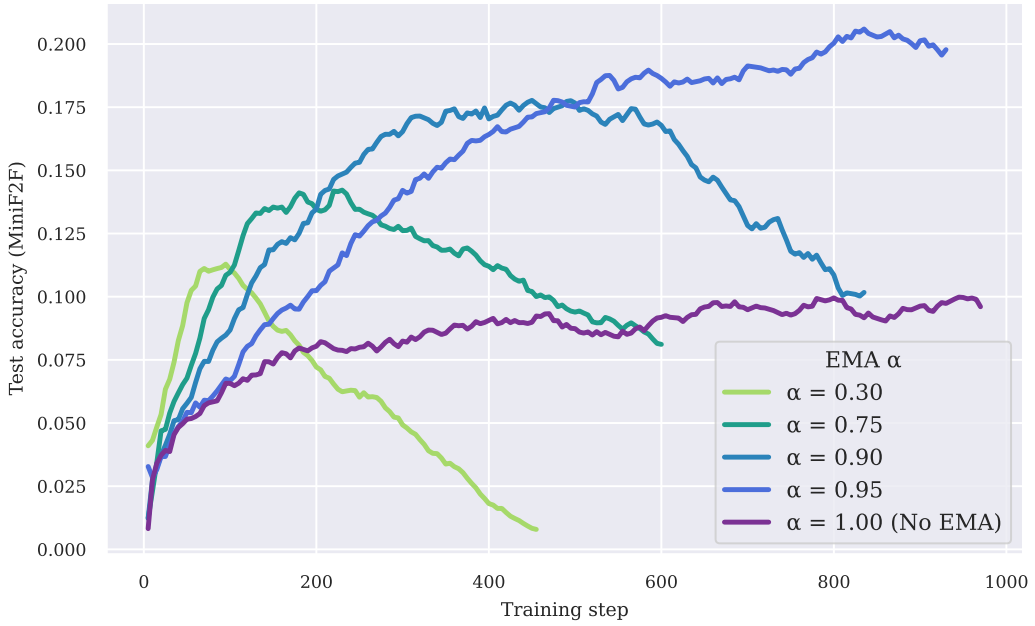

Figure 11: MiniF2F accuracy for Qwen3-8B during training for different values of the EMA interpolation parameter α. More aggressive EMA updates (i.e., lower values of α) accelerate early learning but result in increased instability. The case $\alpha = 1$ corresponds to freezing the teacher weights.

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 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 12 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.

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 13 reports the per-step occurrence counts of three representative epistemic word categories (see Section 5.3) across training runs. When

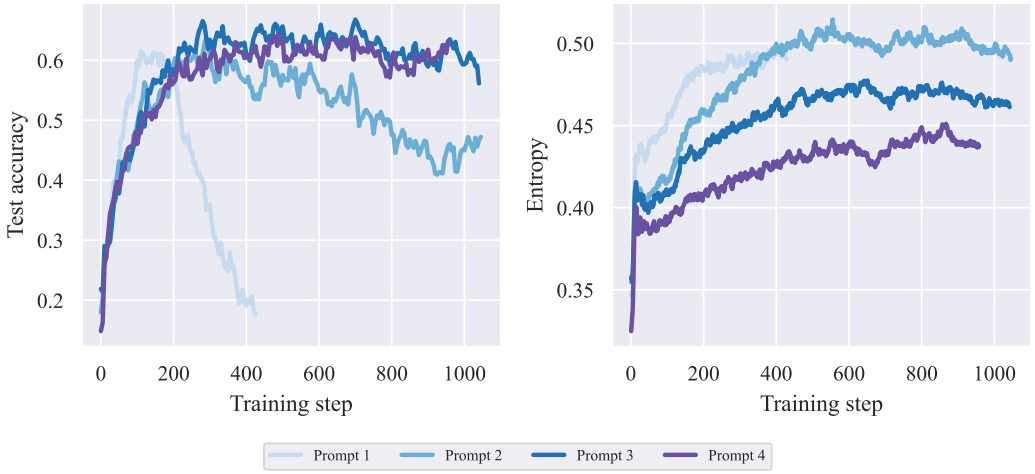

Figure 12: Impact of feedback prompt formulation on training performance. Increasing the level of structure in the prompt improves stability and peak performance.

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.

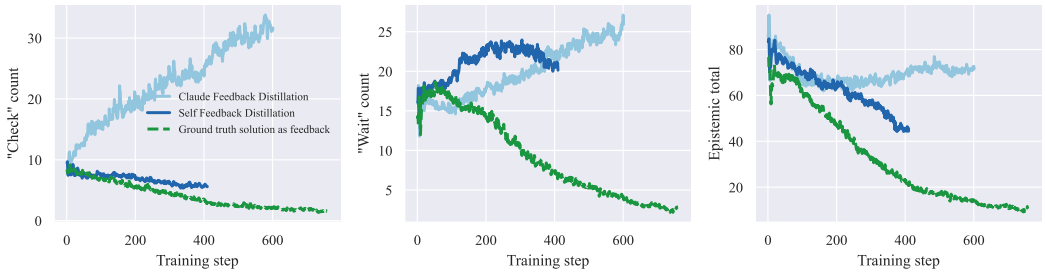

Figure 13: Occurrence counts of epistemic words (*"check"*, *"wait"*, and a combined epistemic total aggregating *wait, hmm, perhaps, maybe, actually, alternatively, seems, might, likely, check*) over training steps for Qwen3-8B. Ground-truth feedback causes a marked decline in all three counts, whereas 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 14), consistent with models that maintain diversity in their generated trajectories rather than converging to short, confident proofs.

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.

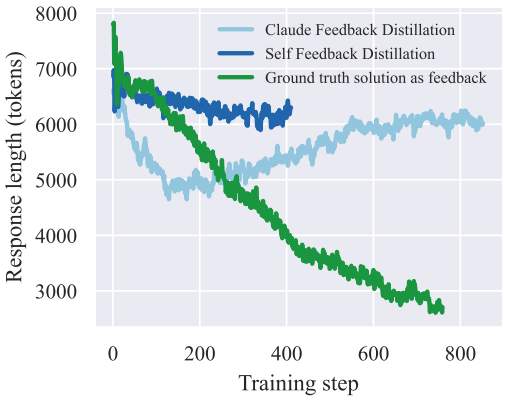

Figure 14: Qwen3-8B mean response length (in tokens) over training steps. Critique-based feedback (Claude and self-feedback) is associated with longer responses than ground-truth feedback, reflecting preserved diversity in generated trajectories.

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 `mul_left_cancel₀` 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.