
LEANADAPT: ADAPTING NEURAL THEOREM PROVERS TO EVOLVING PROOF ENVIRONMENTS

Anonymous authors

Paper under double-blind review

ABSTRACT

Neural theorem provers are trained against a fixed version of their underlying proof library, but libraries evolve: lemmas are renamed, new tactics are added, and proofs written against old APIs fail to compile. Models trained on older versions therefore cannot use new automation and fail on proofs that reference renamed identifiers. We show that Goedel-Prover-V2-8B drops from 83.2% to 76.6% pass@32 on miniF2F when the same proof candidates are verified against an updated Mathlib, with 12.6% of candidates referencing identifiers that no longer exist. We introduce LEANADAPT, a two-stage adaptation pipeline to address evolving proof libraries through LoRA fine-tuning with reasoning traces distilled from a stronger model, followed by GRPO reinforcement learning with model weight averaging. We find that adapting to newer libraries can be beneficial, achieving 211/244 (86.5%, 95% CI [81.4, 90.7]) pass@32 on the newer library version, a +9.9pp improvement over our re-evaluation of the same model on its native version (83.2%) using matched methodology. This remains below larger models (32B: 88.1%) and self-correction pipelines (90.4%) (Lin et al., 2025a). We find that model weight averaging at $\alpha=0.2$ recovers output diversity lost to entropy collapse during RL, bridging from 177/244 (pure RL) to 211/244. We release our adaptation infrastructure for reproducibility, including a cross-verified dataset of 3.4M Lean 4 problems (349K proven theorems, 2.1M RL targets), a Lean 4 verification server, and training and evaluation pipelines.

1 INTRODUCTION

Large language models (LLMs) have been applied to formal theorem proving since they were first tested on formal mathematics with GPT-F (Polu & Sutskever, 2020), and modern systems achieve pass@32 above 80% on standard benchmarks by generating proofs in interactive proof assistants such as Lean 4 (de Moura & Ullrich, 2021). Recent proof-solving systems such as DeepSeek-Prover-V1.5 (Xin et al., 2024b) and Goedel-Prover-V2 (Lin et al., 2025a) have pushed pass@32 accuracy on the miniF2F benchmark (Zheng et al., 2021) beyond 80%.

However, the environments in which proof-solving systems operate are not static. Lean’s standard mathematical library, Mathlib (The mathlib Community, 2020), evolves rapidly to meet the needs of the community, adding new lemmas, tactics, and automation to empower human and AI-assisted automated theorem proving (ATP). Recent Mathlib versions have introduced powerful tactics such as `grind`, a congruence closure engine, which appears in 595 uses across 96 formalized competition problems in the Compfiles repository (Renshaw, 2024), and `positivity` is used in 4 of the 12 solutions for the Putnam 2025 exam from AxiomProver (Axiom Math, 2025). Models trained on older versions cannot access these new tactics, nor even attempt them, as we show in Section F. Closed-source systems such as AxiomProver (Axiom Math, 2025) already operate on current library versions and likely can adapt to newer versions through in-context learning, tool calling, self-correction, and other techniques, but open-weight provers trained on older versions are at a structural disadvantage.

We verify the *same proof candidates* from three state-of-the-art $\leq 8B$ models on both their training-time native Lean version and a version released eleven months later. Goedel-Prover-V2-8B drops from 83.2% to 76.6% pass@32 (−6.6pp), DeepSeek-Prover-V2-7B from 69.7% to 64.3% (−5.3pp), while Kimina-Prover-7B shows no degradation (64.8% on both). We trace the failures to Mathlib API

054 drift, as 12.6% of generated proof candidates reference APIs that have been renamed in the newer
055 version.

056 We propose LEANADAPT, a two-stage adaptation pipeline that recovers performance lost to library
057 evolution and, on miniF2F, surpasses the model’s performance on its original train-time library
058 version.
059

- 060 1. **LoRA Fine-Tuning with Distilled Reasoning.** We train a rank-16 LoRA adapter (Hu
061 et al., 2022) on 8,885 examples: 7,597 verified proofs paired with natural language proof
062 plans distilled from Claude Opus 4.6, plus 1,288 examples containing updated API names.
063 This yields +16 problems over the baseline on the new library version, while preserving
064 chain-of-thought reasoning.
- 065 2. **GRPO Reinforcement Learning with Model Averaging.** We apply Group Relative Policy
066 Optimization (Shao et al., 2024) with binary verification rewards on 106K problems. RL
067 alone suffers from entropy collapse (policy entropy drops from 1.08 to 0.087 in 23 steps),
068 destroying output diversity. We overcome this via model weight averaging (Wortsman et al.,
069 2022a), interpolating between the RL checkpoint and its pre-RL base. Through ablations,
070 $\alpha=0.2$ yields +8 additional problems.

071 LEANADAPT achieves 211/244 (86.5%) pass@32 on miniF2F under the newer library version, a
072 +9.9pp gain over our matched re-evaluation of the base model on v_{old} (83.2%), demonstrating that
073 adaptation to an evolving library can raise performance, not just restore it.
074

075 Our contributions are as follows:

- 076 • We quantify library-version fragility in neural theorem provers: same-candidate evaluation
077 across two Mathlib versions reveals up to 6.6pp degradation from API drift alone.
- 078 • We introduce LEANADAPT, a two-stage pipeline (LoRA with distilled reasoning from
079 Claude Opus 4.6, followed by GRPO RL with model averaging) that achieves 86.5%
080 pass@32 on v_{new} (95% CI [81.4, 90.7]), a +9.9pp gain over our matched re-evaluation on
081 v_{old} (83.2%), solving 46 competition-level problems the baseline could not (Appendix G).
- 082 • We show that model weight averaging at low α overcomes entropy collapse in RL for
083 theorem proving, recovering diversity that RL training destroys. Pure RL achieves only
084 177/244, while averaging yields 211/244. The coefficient α controls a trade-off between
085 pass@1 consistency and pass@32 coverage.
- 086 • We release the complete adaptation infrastructure: a cross-verified Lean 4 dataset (3.4M
087 problems, 349K proven theorems, 2.1M RL targets across 21 datasets), a verification server,
088 and end-to-end training pipelines.
089

090 2 RELATED WORK

091 2.1 NEURAL THEOREM PROVING

092 Prior work in neural theorem proving assumes a fixed proof environment. We study open-weight
093 *adaptation* to an evolving one, tracking a moving Mathlib API surface without the frontier-scale
094 agentic stacks now used by closed systems.

095 Early work by Polu & Sutskever (2020) demonstrated that transformer language models can generate
096 proofs in Metamath, achieving results that were accepted into the main Metamath library. Lample
097 et al. (2022) introduced HyperTree Proof Search (HTPS), an online training procedure inspired by
098 AlphaZero that implemented proof-tree tactic-level search, improving miniF2F accuracy from 31%
099 to 42%.
100

101 Subsequent systems moved toward whole-proof generation via autoformalization (Wu et al., 2022)
102 and large-scale synthetic data: DeepSeek-Prover (Xin et al., 2024a) achieved 52% on miniF2F with
103 8M synthetic statements, and DeepSeek-Prover-V1.5 (Xin et al., 2024b) reached 63.5% by combining
104 RL with Monte-Carlo tree search. Li et al. (2024) survey this area comprehensively.
105

106 Goedel-Prover (Lin et al., 2025b) achieved 57.6% pass@32 on miniF2F through iterative synthetic
107 data generation, and Goedel-Prover-V2 (Lin et al., 2025a) scaled to 84.6% (8B) and 88.1% (32B)

108 via scaffolded data synthesis, self-correction, and model averaging. Kimina-Prover (Wang et al.,
109 2025) pioneered reasoning-driven exploration, reaching 80.7% pass@8192 on miniF2F. DeepSeek-
110 Prover-V2 (Ren et al., 2025) reached 88.9% pass@8192 with a 671B parameter model using subgoal
111 decomposition (82.4% at pass@32; see Table 1). AlphaProof (Hubert et al., 2025) solved IMO
112 problems at silver-medal level through AlphaZero-style reinforcement learning in Lean.

113 Moreover, there exist closed-source theorem-proving agents that leverage frontier models and out-
114 perform existing open-weight models through sheer scale and compute. While their internals are
115 not public, systems such as AxiomProver (Axiom Math, 2025) and Aristotle (Achim et al., 2025)
116 likely combine multi-agent orchestration with frontier models (e.g., GPT-5.2 Pro) under extended
117 chain-of-thought, tool-calling for Lean kernel or Lean LSP feedback, lemma retrieval via search en-
118 gines like LeanExplore (Asher, 2025), recursive subgoal decomposition, and human expert guidance.
119 These closed-source systems have helped prove frontier problems such as Erdős Problem #728 using
120 GPT-5.2 Pro orchestrating Aristotle (Sothanaphan, 2026).

121 122 2.2 REINFORCEMENT LEARNING FOR FORMAL REASONING

123 Reinforcement learning has become a key technique for improving language models in verifiable
124 domains such as coding and mathematics. Proximal Policy Optimization (PPO; Schulman et al.,
125 2017) was the dominant algorithm for RLHF in language models, but Shao et al. (2024) introduced
126 Group Relative Policy Optimization (GRPO), which eliminates the PPO critic by estimating baselines
127 from group scores and reducing memory requirements while achieving strong results on mathematical
128 reasoning benchmarks. GRPO has since been adopted by multiple theorem proving systems, including
129 Goedel-Prover-V2 (Lin et al., 2025a), Kimina-Prover (Wang et al., 2025), and DeepSeek-Prover-
130 V2 (Ren et al., 2025).

131 For theorem proving, the reward signal is binary. A proof either compiles and is verified by the proof
132 assistant, or it does not. The resulting sparse reward signal can cause entropy collapse where the
133 policy converges to a narrow set of proof strategies and loses the output diversity needed for pass@*k*
134 evaluation with large *k*, often needed to tackle difficult problems from IMO, AIME, and Putnam
135 problem sets.

136 All our RL experiments exhibit severe entropy collapse (entropy dropping from ~ 1.1 to ~ 0.09)
137 regardless of entropy bonus coefficients. Rather than modifying the RL algorithm, we propose a
138 complementary post-hoc solution: *model weight averaging* between the RL-trained checkpoint and
139 its pre-RL base. This approach is inspired by the weight interpolation literature discussed below but
140 has not previously been applied to overcome entropy collapse in automated theorem proving with
141 Lean.

142 143 2.3 PARAMETER-EFFICIENT ADAPTATION AND MODEL MERGING

144 **Parameter-efficient fine-tuning.** LoRA (Hu et al., 2022) freezes pre-trained weights and injects
145 trainable low-rank matrices, dramatically reducing the number of trainable parameters while pre-
146 serving model quality. This property is particularly important for chain-of-thought models, where
147 full-parameter fine-tuning risks catastrophic forgetting (Luo et al., 2023) of internalized reasoning
148 patterns. We find that this risk is acute for theorem provers: three independent full-parameter SFT
149 experiments all destroy chain-of-thought reasoning (proof plan generation drops from 100% to 3%),
150 while LoRA preserves it. Knowledge distillation techniques for transferring reasoning capabilities
151 from larger models (Magister et al., 2023) inform our approach of distilling natural language proof
152 plans from a stronger model to train the LoRA adapter.

153
154 **Model weight averaging and merging.** Stochastic Weight Averaging (SWA; Izmailov et al., 2018)
155 demonstrated that averaging SGD iterates finds wider optima with better generalization. WiSE-
156 FT (Wortsman et al., 2022b) showed that linearly interpolating between zero-shot and fine-tuned
157 model weights improves robustness to distribution shifts. Model Soups (Wortsman et al., 2022a)
158 extended this to averaging multiple independently fine-tuned models, improving accuracy without
159 additional inference cost. Ilharco et al. (2023) formalized task arithmetic, showing that task-specific
160 knowledge can be composed through vector operations on model weights. In the RLHF setting,
161 WARM (Ramé et al., 2024b) averaged reward models in weight space to mitigate reward hacking,
and WARP (Ramé et al., 2024a) applied weight averaging at multiple stages of the RLHF pipeline.

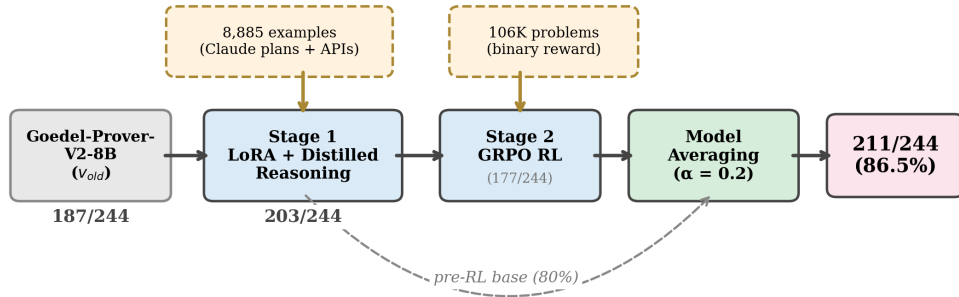


Figure 1: Overview of LEANADAPT. Goedel-Prover-V2-8B (trained on v_{old}) is adapted to v_{new} through (1) LoRA fine-tuning on 8,885 examples with reasoning traces distilled from Claude Opus 4.6 (187→203), and (2) GRPO reinforcement learning followed by model averaging at $\alpha=0.2$ with the pre-RL base (203→211). The final model achieves 211/244 (86.5%) on miniF2F pass@32.

3 METHOD

We present LEANADAPT, a two-stage pipeline for adapting a neural theorem prover trained on Lean v_{old} to a newer version v_{new} where Mathlib APIs have changed. Figure 1 provides an overview.

3.1 PROBLEM SETTING

Let θ denote a neural theorem prover trained on Lean version v_{old} with Mathlib version M_{old} . When M_{old} is updated to M_{new} , API changes cause proofs referencing old identifiers to fail with “unknown identifier” errors. Our goal is to produce an adapted model θ' that maximizes pass@ k on v_{new} .

In our experiments, v_{old} corresponds to Lean v4.9.0-rc1 (April 2024) and v_{new} to Lean v4.21.0 (March 2025), spanning approximately eleven months and 39 API renames. We use Lean v4.9.0-rc1 as v_{old} as it was used by DeepSeek-Prover, Kimina-Prover, and Goedel-Prover, and use v_{new} as Lean v4.21.0 as it was the version used by AxiomProver to solve the Putnam 2025 exam. We evaluate on miniF2F (Zheng et al., 2021) (244 problems) and PutnamBench (Tsoukalas et al., 2024) (672 problems) using pass@ k with the unbiased combinatorial estimator of Chen et al. (2021). Verification uses AXLE (Axiom Math, 2026), a cloud API for Lean 4 proof verification (also used by AxiomProver), in addition to our own custom Lean server, as AXLE only supports Lean versions from Lean v4.21.0 to Lean v4.29.0.

3.2 LARGE-SCALE CROSS-VERIFIED DATASET

Adapting requires training data verified on the target library version. We curate 3.4M Lean 4 problems across 21 datasets from 16 upstream sources, all cross-verified on v_{new} using an automated 4-step pipeline (convert, enrich with verification traces, audit, upload). The conversion step applies cross-version syntax transforms to maximize compatibility. Of these, 349,654 have verified proofs (led by nemotron_proofs with 283K, hf_lean_workbook with 29K, and deepseek_prover with 27K), and 2.1M serve as RL targets (sorry-terminated statements that compile on v_{new}).

3.3 STAGE 1: LORA FINE-TUNING WITH DISTILLED REASONING

We fine-tune Goedel-Prover-V2-8B with LoRA (Hu et al., 2022), a parameter-efficient method that freezes pre-trained weights and injects trainable low-rank matrices into the attention layers.

Training data. We construct 8,885 training examples from our cross-verified dataset: (1) 7,597 verified Lean proofs paired with natural language proof plans distilled from Claude Opus 4.6, and (2) 1,288 examples (upsampled $3\times$, 21.7% of data) containing v_{new} API names. The distilled plans describe the proof strategy, key lemmas, and logical structure, teaching the model both updated APIs and effective proof planning in a single training stage.

Configuration. Rank-16 LoRA with $\alpha_{\text{LoRA}} = 32$, dropout 0.05, targeting q/k/v/o_proj layers (0.19% of parameters). Learning rate 1×10^{-4} with cosine decay, batch size 128, 1 epoch (69 steps), 1h53m on $4 \times \text{H100}$. The merged model achieves 203/244 pass@32 on v_{new} , a gain of +16 over the baseline performance on v_{new} ; ablations against full-parameter SFT and code-only LoRA are reported in Table 3.

3.4 STAGE 2: GRPO REINFORCEMENT LEARNING

We further improve the model using Group Relative Policy Optimization (GRPO; Shao et al., 2024), starting from the LoRA-merged model as the base policy.

Algorithm. For each problem x , GRPO generates $N=8$ candidates $\{y_i\}$ and assigns binary rewards $r_i \in \{0, 1\}$. Advantages are normalized within groups: $\hat{A}_i = (r_i - \bar{r}) / (\sigma_r + \epsilon)$; groups with uniform rewards ($\sigma_r = 0$) contribute zero gradient and are effectively skipped. The policy is updated via a clipped surrogate objective with KL regularization:

$$\mathcal{L}(\theta) = -\mathbb{E}_{x, \{y_i\}} \left[\frac{1}{N} \sum_{i=1}^N \min \left(\rho_i \hat{A}_i, \text{clip}(\rho_i, 1 \pm \epsilon) \hat{A}_i \right) - \beta \text{KL}[\pi_\theta \parallel \pi_{\text{ref}}] \right] \quad (1)$$

where $\rho_i = \pi_\theta(y_i|x) / \pi_{\text{ref}}(y_i|x)$ and $\beta=0.001$.

Reward function and verification safety. The reward is binary: $r_i = 1$ if (a) the response contains a valid `lean4` code block, (b) the extracted proof compiles on our Lean v4.21.0 server, and (c) the proof body contains no `sorry` or `admit`; otherwise $r_i = 0$. We detect sorries by substring match on the proof body after `:= by`.

Our Lean verification server extends the Kimina-Lean-Server (Wang et al., 2025) with a two-layer defense against reward hacking. *Layer 1 (pre-check)*: A source blocklist of 14 regex patterns rejects dangerous constructs before code reaches the REPL, including custom `axiom` declarations, `native_decide` (unsound compiler trust), metaprogramming escape hatches (`run_elab`, `setEnv`, `addDecl`), tactic-suggestion commands (`apply?`, `exact?`) that can leak `sorryAx` via message flooding, and unsafe attributes (`@[implemented_by]`, `@[csimp]`, `@[extern]`). Comment stripping prevents false positives from keywords appearing in comments. *Layer 2 (post-check)*: After compilation succeeds, the server runs `#print axioms` on each declaration and rejects any proof depending on axioms outside the standard allowlist (`propext`, `Quot.sound`, `Classical.choice`). This catches attacks that evade the source blocklist, including hidden `sorryAx` from tactic suggestions and `Lean.trustCompiler` from `native_decide`. Both layers are enabled by default. The blocklist is regex-based and can in principle be evaded by macro obfuscation; kernel-replay verification (e.g., `lean4checker`, `SafeVerify`) would provide stronger guarantees but is too slow for RL reward computation.

Training data and configuration. We use 106K problems (40% easy, 40% medium, 20% hard; zero overlap with evaluation benchmarks). Decontamination uses 13-word n -gram Jaccard similarity (following the n -gram overlap approach common in code decontamination (Chen et al., 2021)) against miniF2F, PutnamBench, and Putnam 2025; the maximum observed similarity between any RL-training statement and any evaluation-benchmark statement is 0.098, well below the 0.3 flagging threshold (99th percentile: 0.070). Additionally, all evaluation benchmark datasets are excluded by name from the RL training pool. Key hyperparameters: learning rate 1×10^{-5} , KL coefficient 0.001, entropy coefficient 0.0005, batch size 64, $N=8$ rollouts, $4 \times \text{H100}$ with FSDP.

Entropy collapse. RL exhibits severe entropy collapse: policy entropy drops from 1.08 to 0.087 over 23 steps, with 80% lost by step 5. The model converges to a narrow tactic repertoire (`nlinarith`, `norm_num`, `ring`, `omega`) and produces shorter proofs.

3.5 MODEL AVERAGING

To recover diversity destroyed by entropy collapse, we average the RL checkpoint with its pre-RL base: $\theta_{\text{final}} = (1 - \alpha) \cdot \theta_{\text{base}} + \alpha \cdot \theta_{\text{RL}}$. We select step 5 (entropy still 0.209) and sweep $\alpha \in \{0.2, 0.3, 0.4\}$ on

miniF2F pass@32 (we note this constitutes test-set tuning; see Section 4.3 for discussion). At $\alpha=0.2$, the final model achieves 86.5% pass@32 on v_{new} , a +9.9pp gain over our matched re-evaluation on v_{old} (83.2%). Notably, the optimal α differs across benchmarks ($\alpha=0.2$ on miniF2F, $\alpha=0.3$ on PutnamBench with 19/672 vs. 8/672), showing that while the averaging principle transfers, the specific coefficient does not—reinforcing the need for held-out selection in future work.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Base model and benchmarks. Goedel-Prover-V2-8B (Lin et al., 2025a) (Qwen3-8B architecture (Qwen Team, 2025), 8B parameters, trained on v_{old}). Evaluated on miniF2F (Zheng et al., 2021) (244 problems) and PutnamBench (Tsoukalas et al., 2024) (672 problems).

Inference and verification. vLLM (Kwon et al., 2023) on A100-80GB, 32 samples per problem, temperature 1.0 and top- p 0.95; see Appendix C for the complete prompt template. All reported v_{new} results use AXLE (Axiom Math, 2026) for verification with sorry-filtering; v_{old} results use a local Lean REPL. To validate cross-verifier consistency, we verified the baseline’s 7,808 candidates with *both* verifiers: after sorry-filtering, lean-server and AXLE agree on all 4,531 valid candidates and all 3,277 invalid candidates (Cohen’s $\kappa=1.0$), confirming that the cross-version gap is not a verifier artifact.

Training. 4×H100 on Modal cloud with DeepSpeed ZeRO-3 (Rajbhandari et al., 2020), with other experiments and evaluations run on UVA’s high-performance computing cluster. Dataset details are in Table 5 (Appendix B); complete LoRA and GRPO hyperparameters are in Tables 6–7 (Appendix C); compute costs are in Table 8 (Appendix D).

4.2 MAIN RESULTS

miniF2F. Table 1 presents pass@ k results for $k \in \{1, 8, 16, 32\}$ on miniF2F.

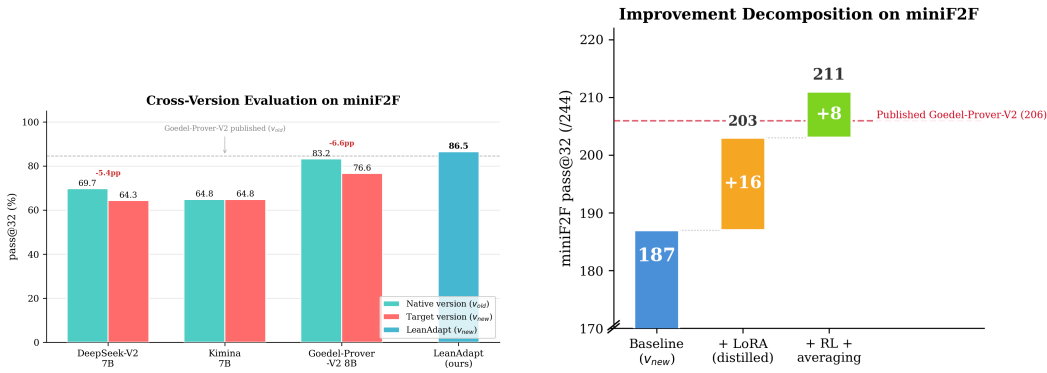
LEANADAPT achieves 86.5% pass@32 (95% CI [81.4, 90.7]) on v_{new} , surpassing all $\leq 8\text{B}$ models on their native v_{old} as well as the larger Kimina-Prover-72B (84.0%) and the 671B DeepSeek-Prover-V2 (82.4% pass@32 whole-proof). Our re-evaluation yields lower pass@32 than published numbers for all three models, likely due to differences in sampling and prompt format; the cross-version gap is measured on the *same* candidates and is thus unaffected. To confirm the library-version gap, we verify the *same candidates* on both versions: Goedel-Prover-V2-8B drops from 83.2% to 76.6% (−6.6pp), DeepSeek-Prover-V2-7B from 69.7% to 64.3% (−5.3pp), and Kimina-Prover-7B shows no degradation (64.8% on both) (Figure 2a). LEANADAPT not only recovers Goedel-Prover-V2’s lost performance but surpasses it: LoRA contributes +16 problems (187→203), and RL with model averaging adds +8 more (203→211) (Figure 2b). LEANADAPT solves 46 new problems but loses 22 the baseline solved (net +24), with most churn originating at the LoRA stage; the lost problems concentrate in easy competition fillers where LoRA’s distilled plans over-complicate direct tactic applications (Appendix G).

Pass@ k profile. Table 2 shows the full pass@ k breakdown, showing how each stage changes the proof strategy distribution. The baseline shows the flat curve characteristic of low-diversity sampling: pass@1 58.0% only climbs to 76.6% at pass@32.

LoRA produces a categorically different sampling distribution. With only an 11.0% per-candidate valid rate (860/7,808), LoRA’s expected pass@1 is 11.0% (the average per-problem c_i/n across the benchmark), scaling steeply to 83.2% at pass@32. The per-problem valid-candidate count peaks at $c=14$ out of 32 (no problem exceeds $c=16$), meaning the model never reliably solves any single problem on the first attempt, yet 203 out of 244 problems have at least one valid proof among 32 samples. We attribute this to the distilled reasoning format: the diverse proof plans from Claude Opus produce high inter-sample variance (each attempt explores a different proof strategy), but only ~48% of responses yield extractable code blocks, dramatically lowering per-candidate reliability. Conditioning on format-valid samples alone, the compile rate rises to ~23%, roughly doubling

Table 1: miniF2F results (244 problems). “Published” uses each model’s native v_{old} . “Our eval” rows use identical inference (32 samples, $T=1.0$), verified with the Lean REPL (v_{old}) and AXLE (v_{new}). Same candidates on both versions confirm the library-version gap. Pass@ k uses the unbiased estimator of Chen et al. (2021). Dashes (“—”) indicate values not reported by the original authors and not reproducible from public artifacts.

Model	Env.	pass@1	pass@8	pass@16	pass@32
<i>Published results ($\leq 8B$ parameters, on v_{old})</i>					
DeepSeek-Prover-V2-7B (Ren et al., 2025)	v_{old}	58.6%	—	—	75.6%
Kimina-Prover-7B (Wang et al., 2025)	v_{old}	61.1%	—	—	78.3%
Goedel-Prover-V2-8B (Lin et al., 2025a)	v_{old}	—	—	—	84.6%
<i>Our eval on v_{old} (Lean REPL verified)</i>					
DeepSeek-Prover-V2-7B	v_{old}	55.4%	66.0%	67.9%	69.7%
Kimina-Prover-7B	v_{old}	50.3%	59.7%	62.0%	64.8%
Goedel-Prover-V2-8B	v_{old}	62.8%	78.8%	81.3%	83.2%
<i>Same models on v_{new} (AXLE verified)</i>					
DeepSeek-Prover-V2-7B	v_{new}	51.1%	61.4%	63.1%	64.3%
Kimina-Prover-7B	v_{new}	51.5%	60.4%	62.6%	64.8%
Goedel-Prover-V2-8B	v_{new}	58.0%	72.0%	74.6%	76.6%
<i>LEANADAPT (Goedel-Prover-V2-8B, on v_{new})</i>					
+ LoRA (distilled reasoning)	v_{new}	11.0%	49.1%	66.0%	83.2%
+ RL + avg ($\alpha=0.3$)	v_{new}	18.9%	57.0%	70.5%	84.0%
+ RL + avg ($\alpha=0.2$, Ours)	v_{new}	4.1%	53.3%	70.5%	86.5%
<i>Published results ($> 8B$ parameters, on v_{old})</i>					
DeepSeek-Prover-V2-671B (Ren et al., 2025)	v_{old}	—	—	—	82.4%
Kimina-Prover-72B (Wang et al., 2025)	v_{old}	—	—	—	84.0%
Goedel-Prover-V2-32B (Lin et al., 2025a)	v_{old}	—	—	—	88.1%



(a) Cross-version pass@32. Same candidates on both v_{old} and v_{new} . Goedel-Prover-V2-8B drops $-6.6pp$; Kimina-7B shows zero degradation.

(b) Improvement decomposition. LoRA contributes $+16$ and RL+avg adds $+8$. Dashed line: Goedel-Prover-V2-8B published v_{old} result (84.6%) for reference.

Figure 2: miniF2F pass@32 results: (a) library-version gap across three models, and (b) LEANADAPT improvement breakdown.

pass@1; whether a distilled-plan prover can be made uniformly format-reliable without collapsing the pass@32 diversity that drives our 83.2% result is an open question.

Pure RL (from an earlier 19.8K-data configuration; see Table 9) inverts this: pass@1 rises to 29.5% but pass@32 falls to 72.5%, reflecting entropy collapse that narrows the search space.

The averaging coefficient α controls a trade-off between single-shot consistency and diversity needed to solve difficult problems. At $\alpha=0.2$, more base diversity is preserved, yielding the best pass@32 (86.5%) at the cost of lower pass@1 (4.1%). At $\alpha=0.3$, more RL signal is retained, producing a balanced profile: pass@1 rises to 18.9% while pass@32 remains competitive at 84.0%.

378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431

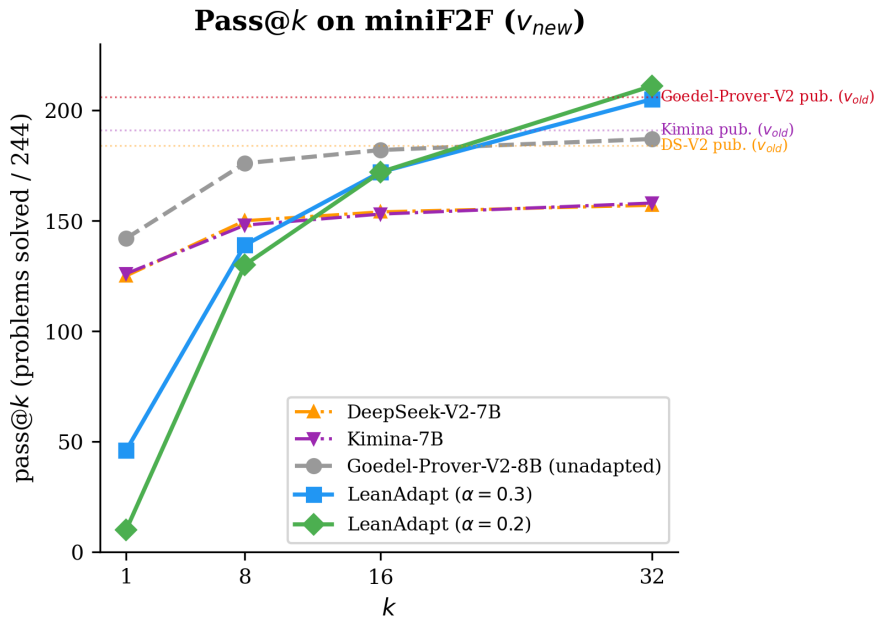


Figure 3: Pass@ k curves on miniF2F (v_{new}). DeepSeek-V2-7B and Kimina-7B plateau at $\sim 158/244$, while LEANADAPT continues scaling to 205–211. Dashed lines: published pass@32 on v_{old} for reference.

Table 2: Expected pass@ k on miniF2F (244 problems, v_{new}), using the unbiased estimator of Chen et al. (2021). Model averaging at $\alpha=0.2$ achieves the best pass@32 (86.5%) while having lower pass@1 (4.1%); $\alpha=0.3$ offers a more balanced profile. “RL only” uses an earlier RL configuration (19.8K data, step 80) to show the effect of entropy collapse without averaging; the averaged rows use the final configuration (106K data, step 5). The LoRA row exhibits an unusual pass@ k profile where pass@1 (11.0%) scales steeply to pass@32 (83.2%), reflecting high output diversity with low per-candidate reliability (see text).

Model	pass@1	pass@8	pass@16	pass@32
Goedel-Prover-V2-8B (unadapted)	58.0%	72.0%	74.6%	76.6%
+ LoRA (distilled reasoning)	11.0%	49.1%	66.0%	83.2%
+ RL only (no averaging)	29.5%	50.4%	54.1%	72.5%
+ RL + avg ($\alpha=0.3$)	18.9%	57.0%	70.5%	84.0%
+ RL + avg ($\alpha=0.2$, Ours)	4.1%	53.3%	70.5%	86.5%

DeepSeek-V2-7B and Kimina-7B plateau early on v_{new} (pass@8 to pass@32 gains only ~ 3 pp), indicating low output diversity. Since Kimina shows no cross-version degradation, this plateau likely reflects the models’ intrinsic diversity limits, not version drift alone. LEANADAPT continues scaling from pass@8 (130) to pass@32 (211), recovering diversity through model averaging.

PutnamBench. On PutnamBench (672 problems), the best LEANADAPT configuration ($\alpha=0.3$) solves 19/672 pass@32, a +171% improvement over the unadapted baseline on v_{new} (7/672). The published Goedel-Prover-V2-8B result (25/672) is on v_{old} ; verifying the same model’s candidates on v_{new} yields only 7/672 (the version gap accounts for 18 of those 25 problems). LEANADAPT closes most of this gap (7 \rightarrow 19) but does not fully recover to the published level, consistent with the observation that PutnamBench problems are harder and more sensitive to API changes than miniF2F, and a lack of diversity likely degrades performance on the more difficult PutnamBench problems. The optimal α differs from miniF2F ($\alpha=0.3$ vs. $\alpha=0.2$), suggesting harder problems benefit from more RL signal. Full results are in Table 4 (Appendix A).

Table 3: Full SFT vs. LoRA on miniF2F pass@32. “Plan Rate” is the fraction of outputs containing non-empty natural language proof plans. The Goedel-Prover-V2 format avoids the Qwen-3 <think> tag, and teaches the model to reason in natural language before writing its Lean code within a “lean markdown format for extraction.

Method	pass@32	Valid Rate	Plan Rate
Baseline (Goedel-Prover-V2 on v_{new})	187	58.0%	100%
<i>Full-parameter SFT (all weights updated)</i>			
148K examples, 2 epochs	128	18.7%	3%
8.9K examples, 1 epoch	152	24.7%	~15%
<i>LoRA (0.19% of parameters)</i>			
Code-only (no reasoning traces)	130	—	—
Distilled reasoning (Ours)	203	11.0%	100%
Think-tag format variant	154	11.8%	—

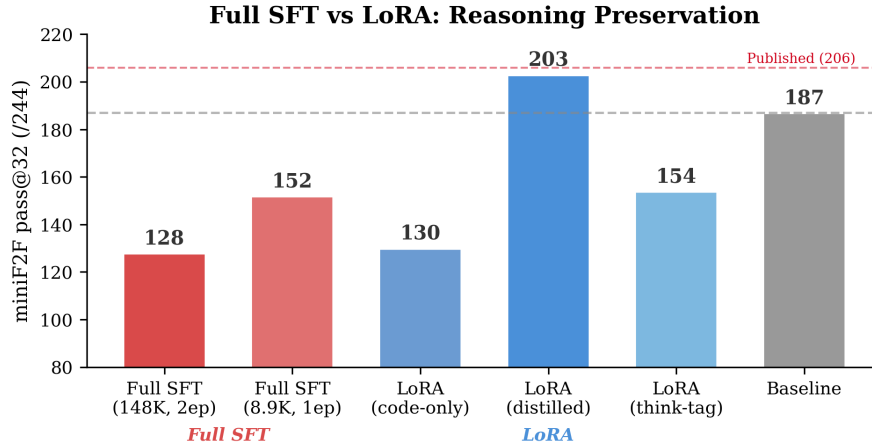


Figure 4: Full-parameter SFT (128, 152) consistently regresses below baseline (187, dashed line), while LoRA with distilled reasoning (203) is the only method that improves.

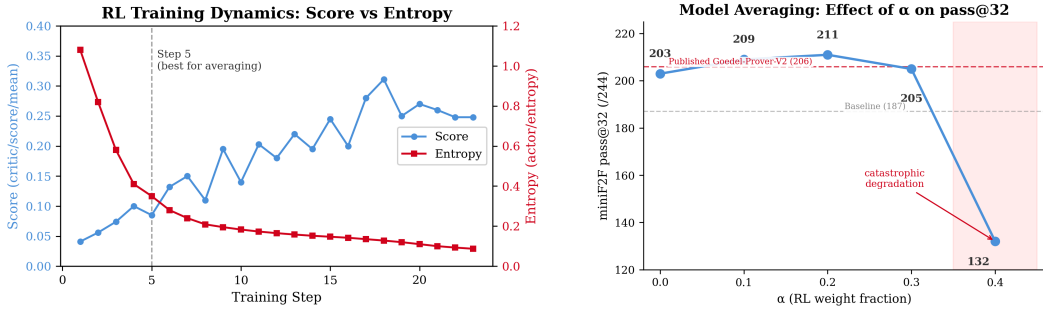
4.3 ABLATION STUDIES

Full SFT vs. LoRA. Table 3 compares fine-tuning approaches. All full-parameter SFT variants regress below baseline (best: 152/244), while only LoRA with distilled reasoning improves (203/244). LoRA trained on code-only data without natural language proof plans also regresses (130/244). The base model generates proof plans in 100% of outputs (median 4,540 chars). After full-parameter SFT on 148K examples, plan generation drops to 3%, per-candidate verification drops from 58.0% to 18.7%, and proofs shrink 4× (median 330 vs. 1,292 chars). Full SFT *did* learn v_{new} APIs (unknown identifier rate: 12.6%→3.0%), but full-parameter updates destroy reasoning as a side effect. LoRA constrains updates to 0.19% of parameters, preserving reasoning (plan rate remains 100%).

Figure 4 visualizes this comparison.

Model averaging coefficient α . For RL, $\alpha=0.2$ is optimal on miniF2F (86.5%); $\alpha=0.4$ catastrophically degrades to 54.1% (McNemar test on paired per-problem outcomes, $p < 0.001$). The difference between $\alpha=0.2$ and $\alpha=0.3$ (84.0%) is not statistically significant (Wilson score 95% CIs overlap: [81.4, 90.7] vs. [78.8, 88.6]), but the cliff at $\alpha=0.4$ is unambiguous. For LoRA, averaging with the pre-LoRA baseline uniformly hurts: this asymmetry reflects that RL averaging uses a *stronger* base (the LoRA-adapted model, 203/244), while LoRA averaging uses a *weaker* one (the unadapted baseline, 187/244). As noted in Section 3.5, α was selected on miniF2F; the fact that $\alpha=0.3$ independently outperforms $\alpha=0.2$ on PutnamBench (19 vs. 8; Table 4) suggests the averaging principle is robust, though the optimal coefficient is benchmark-dependent.

486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539



(a) RL training dynamics over 23 GRPO steps. Score increases while entropy collapses. Dashed line: step 5 (entropy 0.209), selected for averaging. (b) Pass@32 vs. α . Optimum at $\alpha=0.2$ (211/244); sharp cliff at $\alpha=0.4$ (132/244). Dashed line: published v_{old} result (84.6%) for reference.

Figure 5: (a) Entropy collapse during RL training, and (b) sensitivity of pass@32 to the averaging coefficient α .

RL checkpoint and data scale. Earlier checkpoints yield better averaged models: step 5 (211/244) outperforms step 10 (202/244 at $\alpha=0.3$), as earlier steps retain more entropy. Training on 106K problems peaks at 31.1% batch verification rate versus 24% with 19.8K problems, confirming that data scale improves RL.

5 ANALYSIS: ENTROPY COLLAPSE AND MODEL AVERAGING

Figure 5a shows the RL training dynamics. Score increases from 4.1% to 24.8% (peak 31.1% at step 18), but entropy collapses from 1.08 to 0.087. Pure RL achieves only 177/244: the model learns effective tactics but loses diversity needed for pass@32. Model averaging at $\alpha=0.2$ recovers diversity by blending 80% base (diverse reasoning, correct APIs) with 20% RL (learned tactics). Figure 5b shows the sharp sensitivity: $\alpha=0.4$ catastrophically degrades to 132/244 as entropy-collapsed weights dominate.

Appendix F analyzes tactic usage in modern competition proofs, separating API compatibility (addressed by LEANADAPT) from tactic vocabulary expansion (which remains open).

6 CONCLUSION

Neural theorem provers degrade when their training-time Lean library is updated. LEANADAPT is a two-stage pipeline of LoRA fine-tuning on proofs with distilled, followed by GRPO RL with model averaging. LEANADAPT achieves 211/244 (86.5%) pass@32 on miniF2F under v_{new} , a +9.9pp gain over our matched re-evaluation on v_{old} (83.2%), and improves PutnamBench from 7/672 to 19/672 on the same version. We release our dataset, verification server, and training and evaluation pipelines to support future work.

Limitations and future work. Our scope is limited to one base model (Goedel-Prover-V2-8B), one version pair ($v_{old} \rightarrow v_{new}$), single-seed evaluation, and a single closed-source distillation teacher (Claude Opus 4.6). Applying LEANADAPT to additional models, version transition pairs, open-weight teachers for distilled reasoning, and robustly testing multi-seed replication would strengthen the generality claim of LEANADAPT. Moreover, LEANADAPT does not evaluate self-correction (Lin et al., 2025a) after applying LEANADAPT, which could have led to additional performance on miniF2F and PutnamBench. Finally, testing proof generation and self-correction as a baseline for API drift would also provide additional strength to the claim that LEANADAPT is a stronger method than in-context learning from Lean kernel feedback.

540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593

REFERENCES

- Tudor Achim, Alex Best, Alberto Bietti, Kevin Der, Mathis Fédérico, Sergei Gukov, Daniel Halpern-Leistner, Kirsten Henningsgard, Yury Kudryashov, Alexander Meiburg, Martin Michelsen, Riley Patterson, Eric Rodriguez, Laura Scharff, Vikram Shanker, Vladimir Sicca, Hari Sowrirajan, Aidan Swope, Matyas Tamas, Vlad Tenev, Jonathan Thomm, Harold Williams, and Lawrence Wu. Aristotle: IMO-level automated theorem proving. *arXiv preprint arXiv:2510.01346*, 2025. Harmonic AI; combines Lean proof search with informal reasoning and geometry solving.
- Justin Asher. LeanExplore: A search engine for Lean 4 declarations. *arXiv preprint arXiv:2506.11085*, 2025. Hybrid semantic + BM25+ + PageRank search over Lean 4 library declarations.
- Axiom Math. AxiomProver: 12/12 on Putnam 2025. <https://x.com/axiommathai/status/2009682955804045370>, 2025. Announcement of automated theorem-proving agent solving all twelve 2025 Putnam problems in Lean 4.
- Axiom Math. AXLE: Axiom Lean Engine. <https://axle.axiommath.ai/>, 2026. Cloud verification API for Lean 4 proofs.
- Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.
- Leonardo de Moura and Sebastian Ullrich. The Lean 4 theorem prover and programming language. In *Automated Deduction – CADE 28*, pp. 625–635. Springer, 2021.
- Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.
- Thomas Hubert, Rishi Mehta, Laurent Sartran, Miklós Z Horváth, Goran Žužić, Eric Wieser, Aja Huang, Julian Schrittwieser, Yannick Schroecker, et al. Olympiad-level formal mathematical reasoning with reinforcement learning. *Nature*, 651(8106):607–613, 2025. doi: 10.1038/s41586-025-09833-y.
- Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. In *International Conference on Learning Representations*, 2023.
- Pavel Izmailov, Dmitrii Podoprikin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. In *Conference on Uncertainty in Artificial Intelligence*, pp. 876–885, 2018.
- Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, pp. 611–626, 2023.
- Guillaume Lample, Marie-Anne Lachaux, Thibaut Lavril, Xavier Martinet, Amaury Hayat, Gabriel Ebner, Aurelien Rodriguez, and Timothee Lacroix. HyperTree proof search for neural theorem proving. In *Advances in Neural Information Processing Systems*, volume 35, 2022.
- Zhaoyu Li, Jialiang Sun, Logan Murphy, Qidong Su, Zenan Li, Xian Zhang, Kaiyu Yang, and Anima Anandkumar. A survey on deep learning for theorem proving. *arXiv preprint arXiv:2404.09939*, 2024.
- 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. *arXiv preprint arXiv:2508.03613*, 2025a.
- Yong Lin, Shange Tang, Bohan Lyu, Ziran Yang, et al. Goedel-Prover: A frontier model for open-source automated theorem proving. *arXiv preprint arXiv:2502.07640*, 2025b.

594 Yun Luo, Zhen Yang, Fandong Meng, Yafu Li, Jie Zhou, and Yue Zhang. An empirical study of
595 catastrophic forgetting in large language models during continual fine-tuning. *arXiv preprint*
596 *arXiv:2308.08747*, 2023.

597
598 Lucie Charlotte Magister, Jonathan Mallinson, Jakub Adamek, Eric Malmi, and Aliaksei Severyn.
599 Teaching small language models to reason. In *Proceedings of the 61st Annual Meeting of the*
600 *Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 1773–1781, 2023.

601 Stanislas Polu and Ilya Sutskever. Generative language modeling for automated theorem proving.
602 *arXiv preprint arXiv:2009.03393*, 2020.

603
604 Qwen Team. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

605 Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. ZeRO: Memory optimizations
606 toward training trillion parameter models. In *Proceedings of the International Conference for High*
607 *Performance Computing, Networking, Storage and Analysis*, 2020.

608 Alexandre Ramé, Johan Ferret, Nino Vieillard, Robert Dadashi, Léonard Hussenot, Pierre-Louis
609 Cedoz, Pier Giuseppe Sessa, Sertan Girgin, Arthur Douillard, and Olivier Bachem. WARP: On the
610 benefits of weight averaged rewarded policies. *arXiv preprint arXiv:2406.16768*, 2024a.

611
612 Alexandre Ramé, Nino Vieillard, Léonard Hussenot, Robert Dadashi, Olivier Bachem, and Johan
613 Ferret. WARM: On the benefits of weight averaged reward models. In *International Conference*
614 *on Machine Learning*, pp. 42048–42073. PMLR, 2024b.

615 Zhizhou Ren, Huajian Xin, Daya Guo, Zhihong Shao, et al. DeepSeek-Prover-V2: Advancing formal
616 mathematical reasoning via reinforcement learning for subgoal decomposition. *arXiv preprint*
617 *arXiv:2504.21801*, 2025.

618
619 David Renshaw. Compfiles: Formalized competition mathematics in Lean 4. <https://github.com/dwrensha/compfiles>, 2024. Collection of 500+ formalized olympiad problems (IMO, AIME, AMC, Putnam, national competitions).

620
621
622 John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy
623 optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

624
625 Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y K Li, Y Wu,
626 and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language
627 models. *arXiv preprint arXiv:2402.03300*, 2024.

628
629 Nat Sothanaphan. Resolution of Erdős problem #728: A writeup of Aristotle’s Lean proof. *arXiv*
630 *preprint arXiv:2601.07421*, 2026. First Erdős problem reportedly resolved autonomously by an AI
631 system (GPT-5.2 Pro + Aristotle, operated by Kevin Barreto).

632
633 The mathlib Community. The Lean mathematical library. In *Proceedings of the 9th ACM SIGPLAN*
634 *International Conference on Certified Programs and Proofs*, pp. 367–381, 2020.

635
636 George Tsoukalas, Jasper Lee, John Bai, et al. PutnamBench: Evaluating neural theorem-provers on
637 the Putnam mathematical competition. In *Advances in Neural Information Processing Systems*,
638 volume 37, 2024.

639
640 Haiming Wang et al. Kimina-Prover preview: Towards large formal reasoning models with reinforce-
641 ment learning. *arXiv preprint arXiv:2504.11354*, 2025.

642
643 Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes,
644 Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig
645 Schmidt. Model soups: Averaging weights of multiple fine-tuned models improves accuracy
646 without increasing inference time. In *International Conference on Machine Learning*, pp. 23965–
647 23998. PMLR, 2022a.

648
649 Mitchell Wortsman, Gabriel Ilharco, Jong Wook Kim, Mike Li, Simon Kornblith, Rebecca Roelofs,
650 Raphael Gontijo Lopes, Hannaneh Hajishirzi, Ali Farhadi, Hongseok Namkoong, and Ludwig
651 Schmidt. Robust fine-tuning of zero-shot models. In *Proceedings of the IEEE/CVF Conference on*
652 *Computer Vision and Pattern Recognition*, pp. 7959–7971, 2022b.

- 648 Yuhuai Wu, Albert Qiaochu Jiang, Wenda Li, Markus N Rabe, Charles Staats, Mateja Jamnik,
649 and Christian Szegedy. Autoformalization with large language models. In *Advances in Neural*
650 *Information Processing Systems*, volume 35, 2022.
- 651
- 652 Huajian Xin, Daya Guo, Zhihong Shao, Zhizhou Ren, Qihao Zhu, Bo Liu, Chong Ruan, Wenda Li,
653 and Xiaodan Liang. DeepSeek-Prover: Advancing theorem proving in LLMs through large-scale
654 synthetic data. *arXiv preprint arXiv:2405.14333*, 2024a.
- 655
- 656 Huajian Xin, Daya Guo, Zhihong Shao, Zhizhou Ren, Qihao Zhu, Bo Liu, Chong Ruan, Wenda Li,
657 and Xiaodan Liang. DeepSeek-Prover-V1.5: Harnessing proof assistant feedback for reinforcement
658 learning and Monte-Carlo tree search. *arXiv preprint arXiv:2408.08152*, 2024b.
- 659
- 660 Kunhao Zheng, Jesse Michael Han, and Stanislas Polu. MiniF2F: A cross-system benchmark for
661 formal Olympiad-level mathematics. *arXiv preprint arXiv:2109.00110*, 2021.

663 A PUTNAMBENCH RESULTS

664

665 Table 4: PutnamBench results (pass@32). Published results on v_{old} ; LEANADAPT results on v_{new} .

666

667

Model	Env.	pass@32
<i>Published results</i>		
DeepSeek-Prover-V2-7B (Ren et al., 2025)	v_{old}	9
DeepSeek-Prover-V2-671B (Ren et al., 2025)	v_{old}	22
Goedel-Prover-V2-8B (Lin et al., 2025a)	v_{old}	25
Goedel-Prover-V2-32B (Lin et al., 2025a)	v_{old}	43
LEANADAPT ($8B$, v_{new})		
Goedel-Prover-V2-8B (unadapted baseline)	v_{new}	7
+ LoRA (distilled reasoning)	v_{new}	3
+ RL + avg ($\alpha=0.2$)	v_{new}	8
+ RL + avg ($\alpha=0.3$)	v_{new}	19

668

669

680

681 **Error analysis.** Of the baseline’s 7 solved problems, LEANADAPT ($\alpha=0.3$) retains 2 (Putnam
682 1988 B1, 2001 A1) and loses 5 (all from the 1960s era and 2008 A1). In exchange, LEANADAPT
683 newly solves 17 problems spanning 1975–2024, including recent problems (2023 A5, 2023 B1,
684 2024 A2) that require v_{new} API knowledge. The lost problems involve proof strategies (e.g., direct
685 `linarith` on π bounds) that the adapted model’s shifted distribution no longer covers; the newly
686 solved problems tend to use broader tactic combinations learned during RL. Overall, LEANADAPT
687 produces fewer valid candidates than the baseline (25 vs. 68 out of 21,504) but distributes them more
688 broadly across problems (19 vs. 7), consistent with the diversity-over-reliability trade-off observed
689 on miniF2F.

691 B DATASET DETAILS

692

693 Our curated dataset comprises 3,394,310 Lean 4 problems across 21 datasets from 16 upstream
694 sources. Table 5 summarizes the verified proof counts by dataset.

695

696 An additional 2,120,718 problems serve as RL targets: they are sorry-terminated statements that
697 compile on v_{new} but lack verified proofs, providing a natural curriculum for reinforcement learning.

698 C TRAINING AND INFERENCE DETAILS

699

700

701 **Prompt template.** All models are queried with the prompt template from Goedel-Prover-V2’s
DeepSeekCoTHandler (Lin et al., 2025a), identical at training and inference time:

Table 5: Verified proof counts by dataset on v_{new} . Only datasets with >0 verified proofs are shown.

Dataset	Verified Proofs
nemotron_proofs	283,456
hf_lean_workbook	29,208
deepseek_prover	27,126
goedel_pset	9,755
compfiles	67
lean_proofs	15
formal_conjectures	13
putnam2025	12
formalmath	2
Total	349,654

Complete the following Lean 4 code:

```

lean4
{formal_statement_with_sorry}

```

Before producing the Lean 4 code to formally prove the given theorem, provide a detailed proof plan outlining the main proof steps and strategies. The plan should highlight key ideas, intermediate lemmas, and proof structures that will guide the construction of the final formal proof.

For inference, we use vLLM (Kwon et al., 2023) with temperature 1.0, top- p 0.95, max input + output 16,384 tokens (32,768 for miniF2F evaluation to capture long plans), and 32 samples per problem. Tables 6 and 7 list the LoRA and GRPO hyperparameters referenced in Sections 3.3 and 3.4 respectively.

Table 6: LoRA (distilled reasoning) training hyperparameters.

Parameter	Value
Rank	16
α_{LoRA}	32
Dropout	0.05
Target modules	q_proj, k_proj, v_proj, o_proj
Trainable parameters	0.19%
Learning rate	1×10^{-4} (cosine decay)
Effective batch size	128
Epochs	1 (69 steps)
Training data	8,885 examples
Training time	1h 53m on 4×H100

D COMPUTE COSTS

Training ran on Modal cloud (H100) and inference/evaluation ran on UVA’s HPC cluster (A100-80GB). Dollar estimates use list pricing at the time of each run; the total excludes failed or restarted jobs.

E RL TRAINING RUNS

Table 9 summarizes all four RL training configurations and their outcomes.

Table 7: GRPO reinforcement learning hyperparameters.

Parameter	Value
Algorithm	GRPO
Learning rate	1×10^{-5}
KL coefficient	0.001
Entropy coefficient	0.0005
Batch size	64
Rollouts per problem (N)	8
Clip ratio	0.2–0.28
Temperature	1.0
Max tokens (prompt + response)	16,384
Advantage normalization	By group std
Training data	106K problems
GPUs	4×H100, FSDP
Reward	Binary (compiles \wedge no sorry)

Table 8: Approximate compute costs. Training ran on Modal cloud (H100, \$3.94/hr). Inference and evaluation ran on UVA’s HPC cluster (A100-80GB, \$2.49/hr equivalent).

Experiment	Hardware	GPU-hours	Cost
Full SFT (148K, 2 epochs)	4×H100	37.4h	\$147
LoRA (3 runs)	4×H100	14.2h	\$56
Full SFT ablation (8.9K)	4×H100	2.6h	\$10
RL training (4 configs)	4×H100	129h	\$508
Inference (~20 eval sweeps)	1×A100-80GB	173h	\$431
Model merging/averaging	1×A100-80GB	26h	\$65
Total		382 GPU-hours	~\$1,220

The low-LR run failed because 2×10^{-6} produced negligible policy gradients with sparse binary rewards. The W-REINFORCE variant was tested to avoid entropy collapse through scaling positive advantages by $\lambda=0.1$ to push the model toward randomness. This did not demonstrate meaningful performance increases even while entropy grew.

F TACTIC REPERTOIRE AND NEW AUTOMATION

To understand whether the adapted model benefits from new tactics available in v_{new} , we analyzed tactic usage across three sources: what modern competition proofs use, what the model attempts, and what succeeds.

New tactics in modern proofs. We surveyed 297 formalized competition problems in the Compfiles repository (Renshaw, 2024) (IMO, AIME, AMC, national olympiads), 259 of which have verified proofs, together with the 12 Putnam 2025 solutions published by AxiomProver. Post- v_{old} tactics are highly present: `grind` (congruence closure, introduced between v_{old} and v_{new}) appears 595 times across 96 files; `lia` (linear integer arithmetic, unavailable in v_{old}) appears 1,341 times across 132 files. Pre- v_{old} but underutilized tactics are underused by the baseline: `positivity` appears 344 times (102 files) in Compfiles and 30 times across 4 of the 12 Putnam 2025 solutions, while `gcongr` appears 101 times (37 files) in Compfiles and twice in a single Putnam 2025 solution.

What the model attempts vs. what succeeds. We examined all 7,808 baseline proof candidates on miniF2F. The model attempts pre- v_{old} tactics `positivity` (1,359 candidates) and `gcongr` (133 candidates), with 23.4% and 13.5% success rates respectively. However, the model *never once attempts* `grind` (0/7,808 candidates), because this tactic did not exist in its training data. A tactic used 595 times in modern competition proofs is entirely absent from the model’s vocabulary.

Table 9: Summary of RL training configurations. Score peak is the maximum batch-level verification rate during training.

Configuration	LR	Data	Score Peak	Entropy Final	Outcome
Low LR	2e-6	19.8K	2.6%	2.376	Dead (LR too low)
GRPO baseline	1e-5	19.8K	24%	0.159	199 (avg $\alpha=0.3$)
W-REINFORCE	1e-5	19.8K	1.5%	3.847	Failed (entropy grew)
GRPO + scaled data	1e-5	106K	31%	0.087	211 (avg $\alpha=0.2$)

Two adaptation challenges. This analysis separates two distinct challenges. Firstly, *API compatibility*; that is, learning renamed identifiers so existing proof strategies compile, which LEANADAPT solves (39 renames, e.g., `le_div_iff` \rightarrow `le_div_iff0`, `ZMod.nat_cast_self` \rightarrow `ZMod.natCast_self`). Secondly, addressing *tactic vocabulary expansion*, or teaching the model to exploit new automation, has yet to be fully solved. Both challenges require adaptation to the evolving environment, but the current LEANADAPT pipeline addresses only API compatibility. Targeting a more recent library release where `grind` and `lia` are mature, and including proofs that use them in training data, would expand the model’s proof strategy space and likely yield further gains.

G NEWLY SOLVED AND LOST PROBLEMS

Table 10 lists 46 miniF2F problems that LEANADAPT solves but the unadapted baseline cannot, identified by cross-referencing AXLE-verified results across both models. These include 7 IMO problems (1968–2019), 6 AIME problems (1984–1999), and 16 AMC 12 problems, demonstrating that LEANADAPT recovers competition-level proving capabilities lost to library evolution.

Table 10: miniF2F problems newly solved by LEANADAPT (LoRA + distilled reasoning) that the unadapted baseline fails on v_{new} . Total: 46 problems across 5 categories.

Category	Count	Problems
IMO	7	1968 P5, 1974 P3, 1977 P6, 1982 P1, 1992 P1, 2001 P6, 2019 P1
AIME	6	1984 P7, 1988 P8, 1990 P15, 1991 P9, 1995 P7, 1999 P11
AMC 12	16	2001 P5/P21, 2002B P4, 2003A P23, 2019A P12, 2020A P4/P9/P15/P25, 2020B P21, 2021A P8/P9/P12/P19/P22, 2021B P13
MATHD	5	algebra 17/208/320, numbertheory 427/451
Other	12	11 algebra inequalities/identities, 1 number theory (Mersenne primes)

Lost problems. Of the 22 problems the unadapted baseline solves but LEANADAPT loses, the vast majority concentrate in the MATHD_algebra and MATHD_numbertheory categories (together roughly 70% of the lost set), with AMC 12 problems accounting for the remainder. Inspection of the lost candidates reveals a consistent pattern: the baseline routinely produces 25 or more valid proofs out of 32 samples for these problems using one-line direct tactics (`norm_num`, `decide`, a single `ring/omega`), whereas LEANADAPT’s outputs for the same problems begin with an elaborate distilled plan that introduces multi-step case analyses and intermediate lemmas which then fail to discharge the simple arithmetic goal. This matches the `pass@1` regression reported in Section 4.2: the distilled-plan distribution trades reliability on trivial problems for diversity on difficult ones, and the 22 lost problems are the “easy-competition” tail of that trade-off.