LeMMA: Learning Math by Meta-Adaptation

Zeyneb N. Kaya
Stanford University
zeynebnk@stanford.edu

Nick Rui
Stanford University
nickrui@stanford.edu

Sandra Yang
Stanford University
aleyang@stanford.edu

1 Introduction

Automated mathematical theorem proving in formal languages has emerged as a critical benchmark for evaluating LLM reasoning capabilities (Yang et al., 2024). Unlike natural language tasks, formal proofs in functional programming languages like Lean 4 (de Moura and Ullrich, 2021) offer objective, mechanically verifiable evaluation, eliminating the ambiguity inherent in assessing informal reasoning. However, progress is severely bottlenecked by data scarcity. Formal proof datasets remain orders of magnitude smaller than code or text corpora, as expert-curated formal proofs are inherently limited in supply and expensive to produce (Yang et al., 2024).

Existing approaches synthetically generate training data offline, creating static datasets that are applied uniformly across all problems (Hubert et al., 2025; Ren et al., 2025). However, different theorems require different background knowledge and reasoning patterns. A model tasked with generating a topology proof, for example, may not benefit from training on synthetic data about number theory. We hypothesize that generating targeted, problem-specific synthetic data at test time—adapting the training distribution to each individual theorem—will more effectively address the model's specific knowledge gaps.

We introduce **LeMMA** (**Le**arning **M**ath by **M**eta-**A**daptation), a framework that reframes theorem proving as a test-time meta-learning task. The input to our system is a formal theorem statement in Lean 4 syntax. The model first generates related synthetic lemmas and proofs. A specialized Weigher model then scores these examples based on their utility for the target problem. Finally, the prover adapts its parameters on this weighted, problem-specific dataset before outputting a verifiable proof attempt (in Lean 4 syntax). By dynamically constructing a custom curriculum at inference time, LeMMA addresses problem-specific knowledge gaps.

2 Related Work

Formal Reasoning with LLMs. Recent LLM theorem provers have demonstrated significant progress on formal reasoning benchmarks. DeepSeek-Prover-V2 (Ren et al., 2025) achieves state-of-the-art performance by training on large synthetic datasets generated through search-based methods. AlphaProof (Hubert et al., 2025) achieved gold-medal performance at the International Math Olympiad (IMO) through reinforcement learning methods. However, these approaches rely on large-scale offline training and do not adapt to individual problem characteristics at test time.

Self-Generated Curricula. Self-play Theorem Provers (STP) (Dong and Ma, 2025) shows that models can construct their own training data. By conjecturing theorems and attempting proofs iteratively, successfully verified theorems become training data for future iterations. This bootstrapping process allows models to discover and learn from mathematical structures they find tractable, effectively curating their own curriculum. This shows self-generated training data provides meaningful learning signals, implying models need not rely solely on human-curated datasets to improve reasoning.

Meta-learning. Meta-learning trains models to rapidly adapt to new tasks without retraining from scratch. Model-Agnostic Meta-Learning (MAML) (Finn et al., 2017) optimizes initial parameters for fast few-shot adaptation, while Test-Time Training (TTT) (Sun et al., 2020) adapts models at inference by updating weights based upon test inputs. Self-Adapting Language Models (SEAL) (Zweiger et al., 2025) extends this through persistent self-improvement: models iteratively generate content, evaluate downstream effects, and reinforce successful updates.

These approaches collectively demonstrate that language models can generate their own training data, adapt at test time, and weigh examples by utility. However, self-play occurs offline with uniform datasets, while test-time adaptation has not been applied to formal reasoning. We combine these capabilities by performing problem-specific self-play at inference time, generating custom training data for each theorem and dynamically adapting the prover.

3 Dataset

We work with formal theorem-proof pairs in Lean 4 from Lean Workbook (Ying et al., 2024) and miniF2F (Zheng et al., 2022), datasets consisting of problems spanning algebra, number theory, calculus, geometry, and combinatorics at the high school to undergraduate level.

We use 4,417 examples from Lean Workbook to build a cold-start training dataset to enable conjecture generation (following Dong and Ma (2025)), 16 examples from miniF2F-train for meta-training of the bilevel optimization loop, and 100 randomly sampled problems from miniF2F-test for evaluation. We use DeepSeek-Prover's standard tokenizer. No additional normalization or augmentation is applied.

Theorem

```
theorem sum_first_n (n : Nat) :
  2 * (sum i in Finset.range (n + 1), i) = n * (n + 1) := by
```

Proof

```
induction n with
| zero => simp
| succ n ih =>
  rw [Finset.sum_range_succ, mul_add, ih]
  ring
```

Figure 1: **Example Lean 4 Code**. This is a proof for the equality $2\sum_{i=1}^{n} i = n(n+1)$.

4 Methods

We introduce LeMMA, which approaches automated theorem proving as a bilevel meta-learning problem. Meta-learning trains models to "learn how to learn" by optimizing for rapid adaptation to new tasks rather than training once on fixed data (Finn et al., 2017). In our setting, each theorem T is treated as a distinct task, and we meta-learn which synthetic training data enables effective adaptation to each specific problem. Given target theorem T, LeMMA generates synthetic auxiliary problems and proofs, learns to weight these examples by their utility for proving T, and adapts model parameters on this weighted dataset before attempting the proof (see Figure 2).

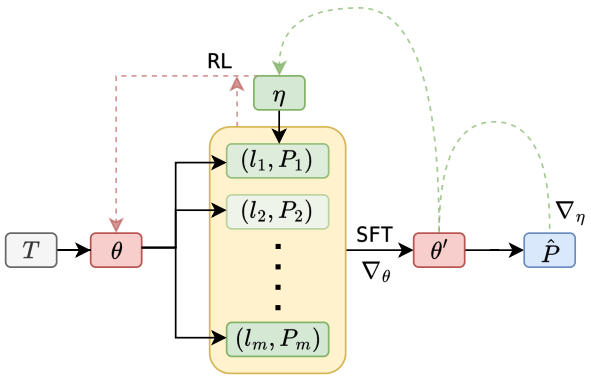

Figure 2: **LeMMA pipeline.** Meta-adaptation workflow from input theorem T to proof attempt $\hat{P}$.

We formulate this as a bilevel optimization problem. We seek parameters that minimize an outer loss on the target theorem while adapting via a weighted update on synthetic auxiliary data:

$$\min_{\eta,\theta} \mathcal{L}_{\text{outer}}(\theta'(\eta,\theta);T) \quad \text{s.t.} \quad \theta' = \theta - \alpha \sum_{i=1}^{m} w_i(\eta;T,l_i,p_i)\nabla_\theta \ell(l_i,p_i;\theta),$$

where $\mathcal{L}_{\text{outer}}$ is the cross-entropy loss on verified proofs of T, θ' are the adapted parameters, α is the inner-loop learning rate, and $\ell(l_i,p_i;\theta)$ is the supervised fine-tuning loss on (l_i,p_i).

In the outer loop, given T, model θ generates m candidate conjecture theorems $\{l_1,...,l_m\}$ related to T. For each conjecture l_i, it attempts proof p_i, creating dataset $\mathcal{A}=\{(l_1,p_1),...,(l_m,p_m)\}$ of auxiliary training examples. Only pairs verified by the Lean 4 compiler are retained. A compact Weigher model η then scores each example's relevance to T by taking (T,l_i,p_i) as input and outputting a scalar score, similar to Calian et al. (2025). These scores are normalized via softmax to obtain a probability distribution over examples: $w_i=\exp(\eta(T,l_i,p_i))/\sum_{j=1}^{m}\exp(\eta(T,l_j,p_j))$.

In the inner loop, rather than fine-tuning all model parameters (expensive for 7B models), we use Low-Rank Adaptation (LoRA) (Hu et al., 2021), which freezes the base model weights and learns low-rank update matrices $\Delta W=BA$ where $B\in\mathbb{R}^{d\times r}$ and $A\in\mathbb{R}^{r\times d}$ with rank $r\ll d$. During forward passes, activations are computed as $h=W_0x+BAx$ where W_0 are frozen pre-trained weights. This reduces trainable parameters significantly while preserving adaptation quality. We train the LoRA adapter via weighted supervised fine-tuning with loss $\mathcal{L}_{\text{inner}}=\sum_{i=1}^{m}w_i\cdot\ell(l_i,p_i;\theta)$, where ℓ is standard cross-entropy for next-token prediction. After adaptation, θ' generates k proof attempts for T, each validated via Lean 4 compiler.

To train the Weigher η, we compute gradients of the outer loss with respect to η. Since η affects θ' through the weights w_i in the inner loop, this requires differentiating through the optimization process itself. We use mixed-mode bilevel differentiation (Kemaev et al., 2025), which combines forward-mode and reverse-mode automatic differentiation to efficiently compute second-order gradients:

$$\frac{\partial\mathcal{L}_{\text{outer}}}{\partial\eta}=\frac{\partial\mathcal{L}_{\text{outer}}}{\partial\theta'}\frac{\partial\theta'}{\partial\eta},\qquad\frac{\partial\theta'}{\partial\eta}=-\alpha\sum_{i=1}^{m}\frac{\partial w_i}{\partial\eta}\nabla_\theta\ell(l_i,p_i;\theta).$$

The quantity $\frac{\partial\mathcal{L}_{\text{outer}}}{\partial w_i}$ indicates whether increasing weight w_i helps or hurts performance on T after adaptation, providing a learning signal for the Weigher.

Beyond updating the Weigher, we also improve the base model θ's ability to generate useful conjectures using Group Relative Policy Optimization (GRPO), a policy gradient reinforcement learning method. The reward for generating conjecture l_i is derived from the meta-gradient: we assign positive reward to examples with negative $\frac{\partial\mathcal{L}_{\text{outer}}}{\partial w_i}$ (examples that reduce the outer loss). We additionally perform standard supervised fine-tuning on all verified synthetic proofs to maintain proof generation quality. For each meta-training theorem, we perform one outer loop iteration: generate synthetic examples, weight them, adapt on weighted data, attempt to prove T, compute meta-gradients, and update both η and θ. The LoRA adapter is reset for each new theorem. This process occurs both during meta-training (to learn η) and at test time (using learned η to adapt to new theorems).

5 Experiments

5.1 Experimental Setup

Base Model and Hyperparameters. We work with DeepSeek-Prover-V2-7B (Ren et al., 2025) as our base model θ, and work with a lightweight LoRA (rank 8, scaling factor $\alpha=8$) throughout training. The Weigher model η follows the base architecture but is compressed to 2 layers with 4 attention heads for efficiency. We generate synthetic theorems and proofs with temperature 0.7 to balance coherence and diversity.

Cold-Start Training. Prover models are only trained to generate proofs. For conjecture generation, we construct a small initialization dataset from a subset of 4,417 problems from Lean Workbook, following the conjecture-training approach of STP (Dong and Ma, 2025). The data is composed of 80% replay proving tasks and 20% theorem generation tasks. Target theorems are identified by selecting lemmas used in seed theorem proofs or theorems sharing common lemmas.

Training. We train LeMMA on 16 theorems from miniF2F-train. For each training theorem, we generate 8 candidate conjectures with 5 proof attempts each, retaining at most 2 verified attempts per conjecture. Meta-gradient computation is expensive with high memory requirements; we use MixFlow-MG reparameterization Kemaev et al. (2025) for mixed-mode forward-over-reverse differentiation and block-level gradient checkpointing for efficient Hessian-vector products. Meta-gradients are computed

over 3 inner loop steps. Both inner-loop SFT and outer-loop meta-updates use learning rate 5×10^{-5}. Due to memory constraints, we use batch size 1 with gradient accumulation to simulate effective batch size 2. We run one outer-loop epoch. We do not perform cross-validation due to our small training set.

Inference. At test time, for each theorem, LeMMA generates 8 conjectures with 2 proof attempts each. We then perform one epoch of unweighted SFT on the filtered synthetic dataset before attempting the target theorem 5 times. Each proof attempt is validated through a Lean 4 REPL (read-eval-print-loop) pipeline.

5.2 Results

We evaluate on a subset of 100 theorems randomly sampled from miniF2F-test (Zheng et al., 2022), which contains 244 formalized olympiad and undergraduate problems. Our primary metric is Lean-verified pass@5 accuracy, which is the percentage of problems where at least one of 5 generated proofs passes full Lean compiler verification.

Table 1 shows our main results. Under a matched 5-sample budget, LeMMA achieves 33% accuracy compared to 23% for DeepSeek-Prover-V2-7B baseline. However, LeMMA incurs approximately $2.2\times$ the FLOPs per theorem due to synthetic data generation, Weigher evaluation, and adapter fine-tuning. To ensure fair comparison, we scale the baseline sample budget to 12 (≈ 2.2 times the compute), yielding 25% accuracy, which LeMMA still outperforms by 8%.

Method	**Sample budget**	**miniF2F-test**
DeepSeek-Prover-V2-7B (Ren et al., 2025)	5	23%
	12	25%
LeMMA	**5**	**33%**

Table 1: Lean-verified accuracy on 100 theorems from miniF2F-test.

5.3 Analysis

We observe a consistent increase in synthetic example verification rates across adaptation steps (Figure 3). Early in training, around 0-3 generated synthetic examples pass verification, whereas later, more than half of the model's generated theorem proof-pairs are valid. Inspecting successful cases, LeMMA frequently learns to introduce short auxiliary lemmas that isolate a key algebraic identity or rewrite step that relate to the main proof.

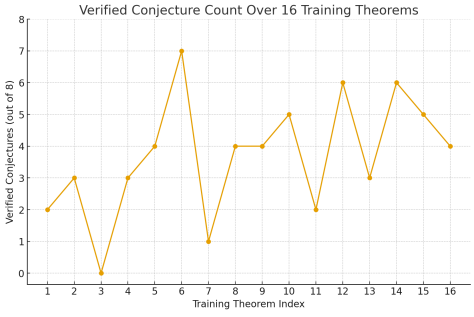

Figure 3: **Progress over training.** Number of successfully verified synthetic theorem-proof pairs (out of 8 generated) across 16 meta-training theorems.

For example, Figure 4 illustrates LeMMA's successful performance on IMO 1983 P6 (an example from miniF2F-test), which asks to prove $a^2b(a-b)+b^2c(b-c)+c^2a(c-a)\geq 0$ for triangle side lengths $a,b,c\in\mathbb{R}$. We observe a model-generated training example that proves the simpler two-variable analog $a^2(a-b)+b^2(b-a)\geq 0$ for nonnegative $a,b\in\mathbb{R}$.

The synthetic two-variable inequality proof utilizes a particular algebraic maneuver: expand the expression, reorder terms, and isolate a nonnegative structure. Later, when attempting the full three-variable cyclic inequality, the model adopts a similar "expand and regroup" approach. While we cannot draw a direct conclusion about the internal mechanisms, the behavior is consistent with LeMMA's design objective: by fine-tuning on problem-adjacent synthetic examples, we hypothesize that the model may shift its internal scoring of candidate proof tokens or intermediate expressions toward patterns more contributive to successful proofs.

From analyzing failed proof attempts, LeMMA struggles with dense and syntactically complex theorems. We observe the lowest pass rates on combinatorics problems, aligning with findings from

Generated Example at Test Time

```
theorem cyclic_inequality (a b : ℝ) (h₀ : 0 < a ∧ 0 < b) :
0 ≤ a^2 * (a - b) + b^2 * (b - a) := by
have h₁ : 0 ≤ a^2 * (a - b) + b^2 * (b - a) := by
  nlinarith [sq_nonneg (a - b), mul_pos h₀.1 h₀.2, sq_nonneg (a + b), sq_nonneg (a - b)]
exact h₁
```

Generated Proof (Verified)

```
theorem imo_1983_p6 (a b c : ℝ)
(h₀ : 0 < a ∧ 0 < b ∧ 0 < c) (h₁ : c < a + b) (h₂ : b < a + c) (h₃ : a < b + c) :
0 ≤ a^2 * b * (a - b) + b^2 * c * (b - c) + c^2 * a * (c - a) := by
have h₄ : 0 ≤ a^2 * b * (a - b) + b^2 * c * (b - c) + c^2 * a * (c - a) := by
  nlinarith [
    sq_nonneg (a - b), sq_nonneg (b - c), sq_nonneg (c - a),
    mul_pos h₀.1 h₀.2.1, mul_pos h₀.2.1 h₀.2.2, mul_pos h₀.2.2 h₀.1,
    mul_pos (sub_pos.mpr h₁) (sub_pos.mpr h₂), mul_pos (sub_pos.mpr h₂) (sub_pos.mpr h₃),
    mul_pos (sub_pos.mpr h₃) (sub_pos.mpr h₁), mul_pos (sub_pos.mpr h₁) (sub_pos.mpr h₂),
    mul_pos (sub_pos.mpr h₂) (sub_pos.mpr h₃), mul_pos (sub_pos.mpr h₃) (sub_pos.mpr h₁),
    sq_nonneg (a - b + c), sq_nonneg (b - c + a), sq_nonneg (c - a + b),
    mul_nonneg (sub_nonneg.mpr h₁.le) (sub_nonneg.mpr h₂.le),
    mul_nonneg (sub_nonneg.mpr h₂.le) (sub_nonneg.mpr h₃.le),
    mul_nonneg (sub_nonneg.mpr h₃.le) (sub_nonneg.mpr h₁.le),
    sq_nonneg (a - b - c), sq_nonneg (b - c - a), sq_nonneg (c - a - b)
  ]
exact h₄
```

Figure 4: **Test-time adaptation on IMO 1983 P6.** LeMMA first generated a simpler proof and, after fine-tuning on that datapoint, successfully proved the original problem. We observe similarities in proof structure and use of Lean tactics, suggesting the model learned a generalizable proof strategy.

Liu et al. (2025), which cite combinatorics as the most challenging domain for LLM provers due to lack of broadly-applicable proof strategies between problems.

5.4 Discussion

In formal theorem proving, test-time compute budgets are typically large: prior work samples over 200,000 proof candidates per theorem (Ren et al., 2025; Calian et al., 2025). Rather than increasing sampling, LeMMA more effectively uses this compute on generating problem-specific synthetic data and performing meta-updates to learn from it. We have shown that within the same budget, test-time compute can be used more effectively through meta-adaptation rather than scaling sample budget.

With only 16 meta-training theorems, overfitting is a serious concern. We attempted to mitigate overfitting by (1) drawing from strictly disjoint datasets (miniF2F-train and miniF2F-test), (2) using low-rank LoRA adapters that are reset for each theorem, and (3) limiting meta-training to a single outer-loop epoch with only three inner adaptation steps. Given these constraints and the fact that hyperparameters were not tuned on the test set, we believe LeMMA is likely not significantly overfit.

6 Conclusion / Future Work

Our results provide initial evidence that allocating test-time compute to meta-adaptation yields better returns than scaling sample budgets. By fine-tuning on self-generated, problem-specific synthetic data, LeMMA achieved 33% pass rate on miniF2F, outperforming a compute-matched DeepSeek-Prover-V2 baseline by 8%. Qualitative analysis suggests that simple synthetic proofs can guide the model toward correct final proofs.

We present LeMMA as a proof of concept. Due to the high memory cost of meta-gradient computation, our training was limited to a small set of data. While this small scale validates the signal provided by our bilevel optimization objective, achieving robust generalization will require scaling to larger training sets and deeper adaptation windows.

Future work could scale to more training theorems and extend LeMMA beyond formal mathematics to symbolic integration, program verification, and multi-step reasoning to assess whether adaptive synthetic generation generalizes across logical domains. This work suggests meta-learning provides a promising direction for test-time improvement in automated reasoning systems.

References

Dan A. Calian, Gregory Farquhar, Iurii Kemaev, Luisa M. Zintgraf, Matteo Hessel, Jeremy Shar, Junhyuk Oh, András György, Tom Schaul, Jeffrey Dean, Hado van Hasselt, and David Silver. 2025. Datarater: Meta-learned dataset curation.

Kefan Dong and Tengyu Ma. 2025. Stp: Self-play llm theorem provers with iterative conjecturing and proving. *arXiv preprint arXiv:2502.00212*.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. *CoRR*, abs/1703.03400.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models.

Thomas Hubert, Rishi Mehta, Laurent Sartran, Miklós Z. Horváth, Goran Žužić, Eric Wieser, Aja Huang, Julian Schrittwieser, Yannick Schroecker, Hussain Masoom, Ottavia Bertolli, Tom Zahavy, Amol Mandhane, Jessica Yung, Iuliya Beloshapka, Borja Ibarz, Vivek Veeriah, Lei Yu, Oliver Nash, Paul Lezeau, Salvatore Mercuri, Calle Sönne, Bhavik Mehta, Alex Davies, Daniel Zheng, Fabian Pedregosa, Yin Li, Ingrid von Glehn, Mark Rowland, Samuel Albanie, Ameya Velingker, Simon Schmitt, Edward Lockhart, Edward Hughes, Henryk Michalewski, Nicolas Sonnerat, Demis Hassabis, Pushmeet Kohli, David Silver, et al. 2025. Olympiad-level formal mathematical reasoning with reinforcement learning. *Nature*.

Iurii Kemaev, Dan A. Calian, Luisa M. Zintgraf, Gregory Farquhar, and Hado van Hasselt. 2025. Scalable meta-learning via mixed-mode differentiation. *arXiv preprint arXiv:2505.00793*.

Junqi Liu, Xiaohan Lin, Jonas Bayer, Yael Dillies, Weijie Jiang, Xiaodan Liang, Roman Soletskyi, Haiming Wang, Yunzhou Xie, Beibei Xiong, Zhengfeng Yang, Jujian Zhang, Lihong Zhi, Jia Li, and Zhengying Liu. 2025. Combibench: Benchmarking llm capability for combinatorial mathematics.

Leonardo de Moura and Sebastian Ullrich. 2021. 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*, volume 12699 of *Lecture Notes in Computer Science*, pages 625–635. Springer.

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. 2025. Deepseek-prover-v2: Advancing formal mathematical reasoning via reinforcement learning for subgoal decomposition. *arXiv preprint arXiv:2504.21801*.

Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei Efros, and Moritz Hardt. 2020. Test-time training with self-supervision for generalization under distribution shifts. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 9229–9248. PMLR.

Kaiyu Yang, Gabriel Poesia, Jingxuan He, Wenda Li, Kristin Lauter, Swarat Chaudhuri, and Dawn Song. 2024. Formal mathematical reasoning: A new frontier in ai. *arXiv preprint arXiv:2412.16075*.

Huaiyuan Ying, Zijian Wu, Yihan Geng, Jiayu Wang, Dahua Lin, and Kai Chen. 2024. Lean workbook: A large-scale lean problem set formalized from natural language math problems. In *Advances in Neural Information Processing Systems*, volume 37. Datasets and Benchmarks Track.

Kunhao Zheng, Jesse Michael Han, and Stanislas Polu. 2022. minif2f: a cross-system benchmark for formal olympiad-level mathematics. In *International Conference on Learning Representations*.

Adam Zweiger, Jyothish Pari, Han Guo, Ekin Akyürek, Yoon Kim, and Pulkit Agrawal. 2025. Self-adapting language models. *arXiv preprint arXiv:2506.10943*.