PROOF-VERIFIER: ENABLING REINFORCEMENT LEARNING FROM VERIFIABLE REWARDS FOR MATHEMATICAL THEOREM PROVING

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement Learning from Verifiable Rewards (RLVR) has revolutionized mathematical reasoning, enabling models like DeepSeek-R1 and OpenAI-o1 to achieve human-level performance on traditional math tasks where answers are single numbers or equations. However, extending RLVR to mathematical theorem proving remains challenging due to the fundamental verification bottleneck: unlike traditional math tasks, theorem proving generates entire reasoning processes that lack reliable automated verification methods for reward signal generation. In this work, we address this verification bottleneck by introducing PROOF-VERIFIER, the first generative verifier specifically designed to enable RLVR applications in mathematical theorem proving. PROOF-VERIFIER supports both formal and informal language (e.g., natural language) proofs, providing the detailed verification capabilities essential for effective reinforcement learning. To train PROOF-VERIFIER, we develop a formal-to-informal translation pipeline for high-quality synthetic data generation and employ a novel two-stage coarse-grained to fine-grained reward modeling mechanism. Experimental validation demonstrates that PROOF-VERIFIER achieves 93% verification accuracy, enabling reliable reward signals for RLVR applications. We show that PROOF-VERIFIER successfully enables effective test-time scaling (79% win rate in best-of-N sampling and 32% improvement in multi-turn proof refinement), and both single-turn and multi-turn RLVR training, consistently improving LLM-based theorem proving performance. Our work establishes the foundation for applying RLVR methodologies to mathematical theorem proving, extending the recent success of reasoning-enhanced models to this challenging domain.

1 INTRODUCTION

Recently, reasoning-enhanced LLMs such as DeepSeek-R1 (DeepSeek-AI et al., 2025) and OpenAI-o1 (OpenAI et al., 2024) have significantly reduced the performance gap between humans and artificial intelligence on traditional mathematical tasks (Lewkowycz et al., 2022) where the answer is a single number or equation. These methods employ Reinforcement Learning from Verifiable Rewards (RLVR) (Ouyang et al., 2022; Wang et al., 2025b), where reward signals are provided by comparing model outputs with reference answers, training models to generate the extended chain-of-thought reasoning (Wei et al., 2023) required to reach verifiable solutions on challenging benchmarks such as HMMT (HMMT, 2025), MATH-500 (Hendrycks et al., 2021; Lightman et al., 2024) and AIME (MAA, 2025).

Despite these advancements, extending RLVR to mathematical theorem proving remains challenging. Models that claim PhD-level competency continue to struggle with mathematical theorem proving problems at the high school or undergraduate level (Guo et al., 2025; Sheng et al., 2025). The primary bottleneck stems from the fundamental verification challenge: mathematical theorem proving tasks (Polu et al., 2022) require generating entire proof processes rather than single numbers or equations, making automated verification for reward signal generation significantly more complex. Moreover, diverse correct proofs can exist for the same statement, making comparisons with reference answers infeasible, which is a critical limitation for RLVR applications.

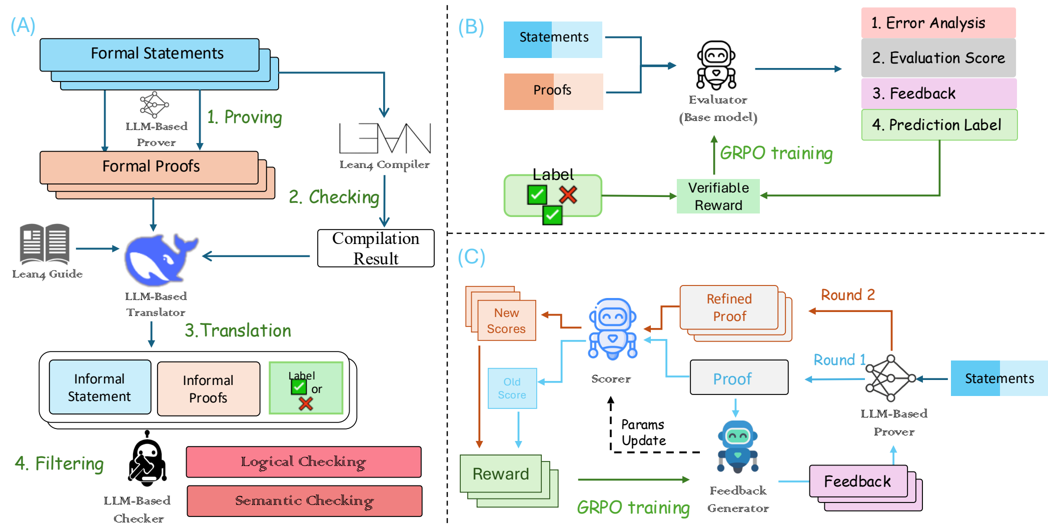


Figure 1: The training framework for PROOF-VERIFIER. (A) Formal-to-informal data synthesis pipeline with ATP verification and filtering. (B) Coarse-grained training with consistency-constrained label prediction. (C) Fine-grained training with proxy rewards from feedback-driven performance improvements.

While Automated Theorem Provers (ATPs) like Lean4 (de Moura et al., 2015; de Moura & Ullrich, 2021) can automatically verify proofs written in formal language by ensuring every deductive step conforms to a foundational logical system, they provide insufficient support for RLVR applications. First, they cannot handle informal language, yet LLMs perform better on natural language due to limited formal language representation in training data. Second, ATP verification results are coarse-grained binary labels that cannot distinguish between minor computational errors and fundamental logical flaws, both receiving the same "incorrect" label without guidance for targeted refinement.

To address this verification bottleneck and enable RLVR applications in mathematical theorem proving, we propose PROOF-VERIFIER, the first generative verifier for mathematical theorem proving tasks that supports both formal and informal language. PROOF-VERIFIER generates a comprehensive verification trajectory containing detailed error analysis, evaluation scores, actionable feedback, and final verification labels for each proof attempt, providing the reliable reward signals essential for effective reinforcement learning.

To support both formal and informal language, we develop a formal-to-informal translation pipeline with controlled generation and balanced labels (Figure 1 A). We sample formal proofs, verify them with ATPs, then translate to natural language with filtering strategies to ensure semantic consistency and logical correctness, achieving reliable data quality verified by human evaluation.

Based on this dataset, we train PROOF-VERIFIER using a novel two-stage coarse-to-fine-grained reward modeling mechanism (Figure 1 B, C). Since only coarse-grained labels are available initially, we design a progressive approach: (1) **Coarse-grained stage**: Label prediction with consistency constraints, where rewards require both accuracy and alignment with majority vote scores, enabling the model to learn robust structured reasoning processes. (2) **Fine-grained stage**: Proxy rewards are estimated by performance improvements brought by generated feedback (pink boxes in Figure 1), where feedback serves as refinement instructions to guide proof revision. This approach enables the model to develop fine-grained discriminative capabilities for distinguishing different error types and providing targeted refinement guidance.

Experimental validation demonstrates that PROOF-VERIFIER achieves 93% verification accuracy on our test set, establishing reliable reward signals for RLVR applications. We show that PROOF-VERIFIER successfully enables effective test-time scaling (Muennighoff et al., 2025), with superior response selection in best-of-N sampling settings (79% win rate) and multi-turn refinement instructions that improve LLM-based prover performance by 32%. Crucially, since PROOF-VERIFIER outputs both evaluation scores (usable as reward signals) and actionable feedback (serving as refinement instructions), it naturally supports both single-turn and multi-turn RLVR training, achieving improved performance in both configurations and demonstrating the successful extension of RLVR methodologies to mathematical theorem proving. Finally, the detailed error analysis generated by PROOF-VERIFIER can be aggregated to help analyze and identify failure patterns for specific models, facilitating targeted improvements in model training iterations.

2 PROOF VERIFIER

2.1 DUAL-LANGUAGE DATASET CONSTRUCTION

Enabling RLVR for mathematical theorem proving requires training data that supports reliable reward signal generation across both formal and informal language proofs. Existing datasets present a critical gap: formal language datasets provide only binary ATP verification without fine-grained feedback, while informal language datasets lack reliable automated verification methods and may introduce validation errors, potentially overlooking logical issues and other subtle errors that are difficult to detect. To address this limitation, we construct a comprehensive dual-language dataset with controlled label quality and balanced coverage using our proposed formal-to-informal translation pipeline.

As shown in Figure 1, LLM-based provers generate 32 proof attempts for each formal statement, with the Kimina Lean Server (Santos et al., 2025) providing rigorous verification labels. DeepSeek-R1 then translates these verified formal statement-proof pairs into corresponding natural language versions. To enhance translation quality, we construct an llm.txt (Howard, 2024) file containing Lean4 syntax, tactics, and common proof methods as context manually. LLM-as-a-Judge is used to subsequently verify semantic consistency and logical correctness to ensure data quality after the conversion process. This formal-to-natural approach is more feasible than natural-to-formal translation, as understanding Lean4 syntax is simpler than generating it. The method ensures proof logic is rigorously compiler-verified before conversion, providing controlled generation of both correct and incorrect proofs with potential errors introduced only during the translation process.

To further enrich our training dataset distribution, we incorporate two additional natural language datasets: the OPC dataset (Dekoninck et al., 2025) provides labeled proof attempts on PutnamBench statements, while RFM Bench (Guo et al., 2025) contributes novel statements spanning different difficulty levels (high school to graduate) and mathematical domains (geometry, algebra, number theory, calculus). Processing details for both datasets are provided in Appendix H and M.

Our final training data comprises these three datasets with balanced sampling, maintaining a 1:1 ratio between natural language and formal language data. All three datasets contribute equally to the natural language data. The training datasets encompass a diverse range of mathematical domains, with detailed statistical distributions presented in Appendix F. For evaluation, we construct out-of-distribution test datasets using different statements and sampling models than those in training, ensuring our evaluation reflects generalization and robustness under distribution shift. The formal language test set contains 1,000 proof attempts with ATP-verified labels, while the natural language test set comprises 100 manually annotated statement-proof pairs with expert human verification.

2.1.1 QUALITY ASSESSMENT

Ensuring translation quality is critical for reliable reward signal generation in RLVR applications. While formal-to-informal translation introduces potential semantic gaps, we demonstrate that systematic filtering using LLM-as-a-Judge can effectively address these challenges. Analysis of 100 randomly sampled translation pairs reveals two primary error categories, which we successfully mitigate through targeted filtering strategies for semantic consistency and logical correctness:

Statement Weakening: Translations occasionally simplify formal statements to less restrictive conditions. We employ LLM-based filtering to detect and remove these cases while preserving valid proof-label pairs, since proofs for stronger claims remain valid for their weaker counterparts.

Syntactic Copying: Complex proofs sometimes result in direct code copying rather than natural language translation. We apply heuristic filtering rules that successfully eliminate these instances, achieving high precision on both validation and out-of-distribution evaluation sets. Detailed case analysis and filtering methodologies are provided in Appendix L.

2.2 Training Approach

2.2.1 Problem Formulation

We formalize the verification task of mathematical proofs as a structured generation task. Given a proof attempt τ and statement x, Proof-Verifier generates a response sequence $r = (a, s, f, p)$ where $a \in \mathcal{A}$ represents error analysis, $s \in [0, 100]$ is the evaluation score, $f \in \mathcal{F}$ denotes feedback, and $p \in \{\text{True}, \text{False}\}$ is the binary correctness judgment. Let $\pi_\theta(r|\tau, x)$ denote our policy parameterized by θ, and $y \in \{\text{True}, \text{False}\}$ be the ground truth label. We optimize θ to maximize expected reward $\mathbb{E}[R(r)]$ under different reward functions $R(\cdot)$ across two training stages, following a coarsed-grained to fine-grained training objective.

2.2.2 Stage 1: Consistency-Constrained Binary Verification

While RLVR training can achieve high accuracy on label prediction tasks, it often leads to high variance in intermediate reasoning chains. Even when models produce correct final judgments, their error analysis and evaluation scoring can be inconsistent across multiple evaluations of the same proof. This inconsistency poses two critical problems: (1) it undermines the model's reliability for fine-grained evaluation tasks where consistent scoring is essential, and (2) it creates unstable training dynamics for Stage 2, which depends on reliable score distributions as reward signals.

To address this challenge, we introduce consistency constraints that enforce both accuracy verification results and consistent evaluation scores. For each input (τ, x), we generate N parallel samples $\{r_1, r_2, \ldots, r_N\}$ and define the correct prediction set as $\mathcal{C} = \{r_i : p_i = y\}$ and score mode: $s_{\text{mode}} = \arg\max_s |\{r_i \in \mathcal{C} : s_i = s\}|$.

Our reward function enforces both accuracy and consistency:

$$R_1(r_i) = 2 \cdot \mathbb{I}[p_i = y \text{ and } s_i = s_{\text{mode}}] - 1 \tag{1}$$

The intuition behind this design leverages the autoregressive generation order where error analysis and scoring precede the final prediction label. By enforcing consistency in the intermediate steps while supervising only the final binary judgment, the model learns to develop stable, coherent reasoning processes that support accurate predictions, which is crucial for the fine-grained capabilities developed in Stage 2. Detailed theoretical analysis demonstrating the convergence properties of this consistency-constrained approach is provided in Section B.1.

2.2.3 Stage 2: Feedback Quality Optimization

Stage 1 enables the model to distinguish correct from incorrect proofs but lacks fine-grained discriminative power to assess varying degrees of proof quality. Stage 2 addresses this limitation by leveraging our sequential generation order where evaluation scores precede feedback. This temporal structure enables mutual supervision: fine-grained error analysis leads to more precise scores, which in turn enables more effective feedback generation.

We initialize both feedback provider F_θ and scorer S_ϕ with Stage 1 parameters: $\theta^{(0)} = \phi^{(0)} = \theta_{\text{Stage1}}$. The training process operates through a multi-step feedback refinement loop. Given a mathematical statement x, an external prover P first generates an initial proof attempt τ_0, which the scorer S_ϕ evaluates to produce a baseline score $s_0 = S_\phi(\tau_0, x)$. The feedback provider F_θ then generates n diverse feedback responses $\{f_1, f_2, \ldots, f_n\}$ based on the initial proof and statement. Each feedback f_i is provided to the prover P, which attempts to incorporate the suggestions and produce a revised proof $\tau_{1,i} = P(\tau_0, f_i, x)$. The scorer evaluates these revised proofs, yielding new scores $s_{1,i} = S_\phi(\tau_{1,i}, x)$ for each feedback-guided revision.

The key insight is that better feedback should lead to improved proofs, as measured by score increases. We therefore define the reward for feedback f_i based on the score improvement it enables:

$$R_2(f_i) = \text{sign}(s_{1,i} - s_0 - \delta) \cdot \mathbb{I}[|s_{1,i} - s_0| > \delta] \tag{2}$$

where $\delta = 10$ filters out minor score fluctuations to focus on meaningful improvements.

To address the instability inherent in jointly optimizing both the feedback provider F_θ and scorer S_ϕ, we employ a momentum encoder strategy (He et al., 2020) with differentiated update frequencies.

The feedback provider parameters are updated at every training step, while the scorer parameters remain frozen for $m = 100$ steps before being updated to match the current feedback provider parameters: $\phi^{(t+1)} \leftarrow \theta^{(t)}$ when $t \bmod m = 0$, and $\phi^{(t+1)} = \phi^{(t)}$ otherwise. Theoretical analysis for this design and the effects of momentum encoder updating strategy for training robustness is provided in Section B.2.

These two stages all use the standard GRPO algorithm for parameters updating:

$$\mathcal{J}_{GRPO}(\theta) = \mathbb{E}\left[\min\left[\frac{\pi_\theta(o|q)}{\pi_{\theta_{old}}(o|q)}\hat{A}, \text{clip}\left(\frac{\pi_\theta(o|q)}{\pi_{\theta_{old}}(o|q)}, 1-\epsilon, 1+\epsilon\right)\hat{A}\right] - \beta\mathbb{D}_{KL}[\pi_\theta||\pi_{ref}]\right] \quad (3)$$

The complete training procedure is summarized in Algorithm 1. Detailed justification and experimental results for using GRPO exclusively without supervised fine-tuning is provided in Appendix C.

Algorithm 1 Two-Stage PROOF-VERIFIER Training

Require: Dataset $\mathcal{D} = \{(\tau_i, x_i, y_i)\}$, prover model P, momentum interval $m = 100$
Initialize $\theta^{(0)}$ with Qwen3-8B parameters
Stage 1: Consistency-Constrained Training
for $t = 1$ to T_1 **do**
    Sample batch $\{(\tau_k, x_k, y_k)\}_{k=1}^{B} \sim \mathcal{D}$
    for each (τ_k, x_k, y_k) in batch **do**
        Generate N responses: $\{r_{k,i} = (a_{k,i}, s_{k,i}, f_{k,i}, p_{k,i})\}_{i=1}^{N} \sim \pi_\theta(\cdot|\tau_k, x_k)$
        Define correct prediction set: $\mathcal{C}_k = \{r_{k,i} : p_{k,i} = y_k\}$
        Compute score mode: $s_{\text{mode},k} = \arg\max_s |\{r_{k,i} \in \mathcal{C}_k : s_{k,i} = s\}|$
        Compute rewards: $R_1(r_{k,i})$ using Equation 1
    end for
    Update via GRPO: $\theta \leftarrow \theta + \alpha\nabla_\theta\mathcal{J}_{GRPO}(\theta)$ using Equation 3
end for
Initialize scorer: $\phi^{(0)} \leftarrow \theta^{(T_1)}$
Stage 2: Feedback Quality Optimization
for $t = 1$ to T_2 **do**
    Sample mathematical statements: $\{x_j\}_{j=1}^{M}$
    for each statement x_j **do**
        $\tau_{0,j} \leftarrow P(x_j)$ {Generate initial proof}
        $s_{0,j} \leftarrow S_\phi(\tau_{0,j}, x_j)$ {Score initial proof}
        Generate n feedback: $\{f_{i,j}\}_{i=1}^{n} \sim F_\theta(\cdot|\tau_{0,j}, x_j)$
        for each feedback $f_{i,j}$ **do**
            $\tau_{1,i,j} \leftarrow P(\tau_{0,j}, f_{i,j}, x_j)$ {Revise proof with feedback}
            $s_{1,i,j} \leftarrow S_\phi(\tau_{1,i,j}, x_j)$ {Score revised proof}
        end for
        Compute rewards: $R_2(f_{i,j})$ using Equation 2
    end for
    Update via GRPO: $\theta \leftarrow \theta + \alpha\nabla_\theta\mathcal{J}_{GRPO}(\theta)$ using Equation 3
    if $t \bmod m = 0$ **then**
        Momentum update: $\phi \leftarrow \theta$ {Transfer knowledge to scorer}
    end if
end for
return $\theta^{(T_2)}$

3 EFFECTIVENESS OF THE PROOF-VERIFIER

In all subsequent experiments, PROOF-VERIFIER uses simple prompts with parallel sampling, as our comparative experiments demonstrates this configuration provides optimal robustness and consistency for evaluation and verification, as detailed in Appendix C.1.

3.1 PERFORMANCE ANALYSIS

Table 1: Performance comparison of different language models on the mathematical proof evaluation task.

Verifier	Natural Language				Formal Language			
	Acc	Prec	Rec	F1	Acc	Prec	Rec	F1
Qwen3-8B	0.57	0.48	0.63	0.54	0.62	0.52	0.68	0.59
Qwen2.5-72B	0.53	0.44	0.58	0.50	0.58	0.48	0.63	0.54
Magistral	0.59	0.50	0.65	0.57	0.64	0.54	0.70	0.61
Gemma	0.58	0.49	0.64	0.56	0.63	0.53	0.69	0.60
Qwen3-235B	0.71	0.62	0.75	0.68	0.76	0.67	0.80	0.73
Deepseek-R1	0.73	0.64	0.76	0.69	0.78	0.69	0.81	0.75
Ours	**0.93**	**0.93**	**0.94**	**0.93**	**0.91**	**0.90**	**0.91**	**0.90**

To verify the reliability of PROOF-VERIFIER for RLVR applications, we first evaluate its verification accuracy and the correlation of fine-grained scores for model ranking tasks.

Verification Accuracy: Table 1 Left presents the verification accuracy results across both language modalities. For natural language proofs, PROOF-VERIFIER significantly outperforms larger open-source models on metrics including accuracy and F1, achieving better correlation with human evaluation, which demonstrates reliable scoring and more consistent alignment with human preferences. The primary error source in existing open-source models is their tendency to incorrectly classify false proofs as correct, exhibiting high recall but low precision, indicating insufficient error detection capabilities. For formal language verification (Table 1 Right), PROOF-VERIFIER achieves superior accuracy compared to other models without access to compilation results, demonstrating the model's ability to better interpret and evaluate formal language proofs. When compilation results are included, PROOF-VERIFIER achieves 0.98 accuracy, with ATP verification serving as the lower bound. Despite this high baseline, PROOF-VERIFIER maintains significant advantages through its generated feedback, which provides substantially greater utility than ATP compilation results. Comparative examples illustrating this advantage are provided in Figure 6 in Appendix.

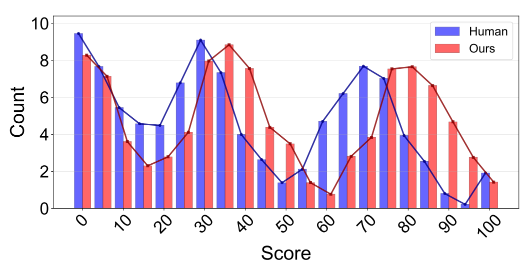


Figure 2: Distribution comparison between human and model evaluation scores using 100 randomly sampled items from the test dataset.

Ranking Correlation: We further validate the reliability of fine-grained evaluation scores by comparing model rankings derived from these scores against ground truth rankings. For natural language proofs, ground truth rankings are derived from human evaluation scores, while formal language rankings are computed using Pass@32 metrics. The Pearson correlations are 0.83 and 0.91 respectively at the individual item level, while completely consistent at the model level. The score distributions of our model and human evaluation are shown in Figure 2, demonstrating the effectiveness of our training method. While PROOF-VERIFIER generates consistently higher scores than human evaluators, the distributions maintain the same trend. This systematic offset reflects LLM-as-a-judge's higher tolerance for errors due to weaker error detection capabilities, but the relative quality rankings remain accurate as shown by strong correlations.

3.2 ABLATION STUDIES

As shown in Table 2, incorporating our synthetic formal-to-informal translated data improves verification performance on both language modalities, demonstrating that controlled label quality and increased data diversity are crucial for reliable reward generation. For the training methodology, the consistency constraint significantly reduces score variance across multiple samples of the same proof-statement pair, providing the stable evaluation capabilities necessary for reliable reference standards in subsequent training stages. Building on this stability foundation, the fine-grained feedback training component increases discriminative power across different proof qualities, as evidenced by improved score distributions that better differentiate between varying proof attempts. To further verify the theoretical feasibility and effectiveness of our proposed Stage 2 method, we employ human evaluators to conduct pairwise comparisons of verification trajectories, where we separately rank the error analysis, actionable feedback, and refined proof attempts from each trajectory.

The correlation between error analysis and feedback rankings is 0.90, while the correlation between feedback and refined proof attempt rankings is 0.95. This high consistency validates our hypothesis that better error analysis leads to better feedback generation, which in turn produces better proof refinements. Details are provided in Appendix D.

Table 2: Ablation study results.

Setting	ACC	ICC	Variance
Data Ablation			
OPC	0.82	0.53	0.13
+ RFM	0.87	0.52	0.11
+ Ours	0.91	0.55	0.14
Method Ablation			
Baseline	0.91	0.55	0.14
+ Consistency Constraint	0.91	0.57	**0.09**
+ Proxy Reward	**0.93**	**0.57**	0.11

3.3 Case Analysis

We evaluated the quality of generated error analysis and actionable feedback using human annotators across both formal and natural language. Human evaluation shows that Proof-Verifier locates approximately 81% of errors when averaged among all annotators, indicating the model learns detailed analysis of the entire proof process to identify errors in each proving step. For feedback, annotators considered 87% of the generated feedback to be useful and actionable, capable of helping prover models correct errors in previous attempts. This demonstrates that Proof-Verifier generates the high-quality feedback essential for effective RLVR applications and suggests potential for performance improvement during inference, which is discussed in Section 4.1. Case examples are provided in Appendix N.

4 Enabling RLVR for Mathematical Theorem Proving

To validate that Proof-Verifier provides reliable reward signals for RLVR applications, we first demonstrate its effectiveness in test-time scaling scenarios, which represent a preliminary but essential validation of our verifier's practical utility for reinforcement learning settings.

4.1 Best-of-N

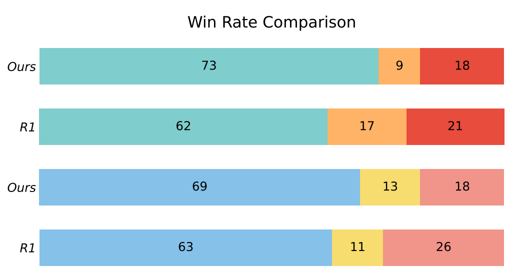


Figure 3: Win rate comparison between models on Natural Language (top two rows) and Formal Language (bottom two rows) tasks. The segments from left to right represent win, tie, and loss rates respectively. R1 means Deepseek-R1-0528.

We employ Proof-Verifier evaluation scores to rank proof attempts from LLM-based provers in best-of-N sampling settings, selecting the highest-scoring attempt. For natural language evaluation, we compare the selected proof against single-sample baselines through human preference evaluation, measuring win rates where best-of-N selections outperform or tie with single samples. For formal language evaluation, we use ATP verification to determine success, with ties occurring when both best-of-N and single-sample approaches pass verification. Experimental results (Figure 3) demonstrate that our model's selected samples achieve higher win rates compared to single samples and outperform the DeepSeek-R1 model on our test dataset. This preliminary validation confirms that Proof-Verifier generates evaluation signals that effectively guide proof selection, establishing the foundation for more comprehensive RLVR applications in subsequent experiments.

4.2 Refinement Based on Feedback

To validate that Proof-Verifier feedback enables iterative improvement essential for multi-turn RLVR applications, we evaluate refinement capabilities across both language modalities. For natural language proofs, we assess whether refined proofs show improvement through human evaluation. For Lean4, we use ATP verification to compare pass@k performance before and after refinement.
Natural Language: Human annotators found that 73% of the feedback effectively identifies errors and provides actionable guidance. However, only 51% of errors were successfully addressed in the refinement process, while 17% of proofs showed minimal changes and 32% introduced new errors.

This gap between feedback effectiveness (73%) and successful error correction (51%) reveals limitations in current LLM-based provers' ability to utilize external guidance, highlighting an important direction for future RLVR training that focuses on improving feedback utilization capabilities. **Formal Language:** For Lean4 formal proofs, refinement improved pass@k performance from 37% to 51%, demonstrating that PROOF-VERIFIER feedback provides valuable guidance for correcting formal proof errors. The feedback primarily addresses: 1) correcting boundary condition assumption errors, 2) clarifying unclear theorem scope, and 3) resolving incomplete proofs that use `sorry` placeholders to skip proof goals. This 14-point improvement demonstrates that feedback-guided refinement achieves better scaling results with fewer sampling attempts compared to sequential and parallel scaling approaches, validating the practical utility of our verifier for RLVR applications.

4.3 REINFORCEMENT LEARNING

4.3.1 SINGLE-TURN & MULTI-TURN RL

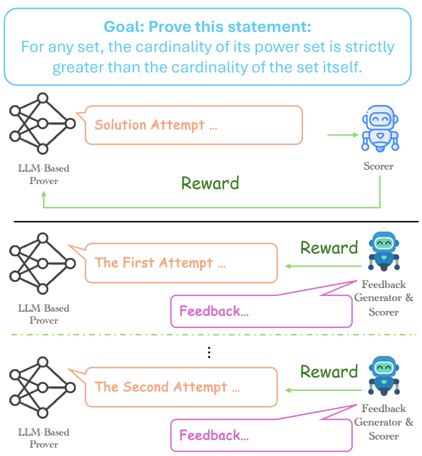


Figure 4: Comparison between single-turn and multi-turn reinforcement learning approaches.

Having demonstrated that PROOF-VERIFIER provides reliable evaluation scores and actionable feedback for test-time applications, we now validate the core claim of our work: that our verifier enables effective RLVR training for mathematical theorem proving. This represents the ultimate test of whether we have successfully addressed the verification bottleneck that previously prevented RLVR applications in this domain. We evaluate PROOF-VERIFIER in reinforcement learning settings under two configurations that reflect different RLVR paradigms (as shown in Figure 4). For training, we use Qwen3-8B for informal language proofs and DeepSeek Prover 2 for formal language proofs. In single-turn RL, PROOF-VERIFIER acts as a reward model, providing binary reward signals based on label predictions (True/-False) for each proof attempt, which validates the basic RLVR functionality where our verifier provides the verifiable reward signals that were previously missing. In multi-turn RL, PROOF-VERIFIER serves dual roles: as a feedback provider that engages in multi-turn conversations by providing refinement instructions, and as a reward model that supplies binary reward signals based on correctness predictions for each round of proof attempts. This configuration demonstrates the full potential of our approach, enabling iterative proof development guided by detailed feedback.

4.3.2 EXPERIMENTAL RESULTS

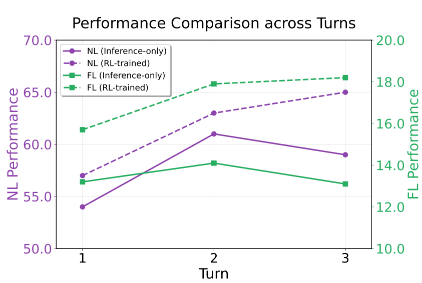


Figure 5: Multi-turn refinement performance comparison.

Our experimental results demonstrate that RL training enables the model to significantly reduce errors and improve output quality compared to the baseline model (Figure 5). Notably, the quality of individual sample generation improved substantially, narrowing the performance gap with best-of-n sampling and achieving higher win rates against reference solutions.

In our multi-turn RL framework, the model receives two complementary signals at each turn: (1) explicit feedback from the verifier, provided as in-context information to guide the prover model, and (2) scalar reward scores from the verifier, quantifying the relative quality of the current turn's proof attempt. The prover model's observation at turn t consists of the complete interaction history $\{p_0, f_0, p_1, f_1, \ldots, p_{t-1}, f_{t-1}\}$, while the verifier focuses solely on the current proof attempt p_t for scoring and feedback generation, where p_i and f_i represent the proof and feedback at turn i, respectively. Crucially, reward signals are attributed only to the current turn's actions, thereby simplifying the credit assignment problem.

Multi-turn RL training yields two improvements: First, the model's capacity to utilize feedback improves, with the gap between feedback quality and actual proof improvement narrowing compared to prompt-based refinement approaches. Second, the performance degradation commonly observed in multi-turn feedback-based systems is mitigated, as the prover becomes more robust in leveraging beneficial feedback while avoiding deterioration from erroneous guidance (as shown in Figure 5). This improvement can be explained by our designed reward structure and credit assignment strategy. By optimizing only the current turn's proof generation regardless of feedback correctness, the model learns to selectively utilize accurate feedback for enhanced proof quality while developing resilience against noisy or incorrect feedback. This approach implicitly trains the model to maintain correct proofs and refine incorrect ones across diverse scenarios, leading to improved robustness in multi-turn interactions.

5 Related Work

Mathematical theorem proving has seen significant progress with benchmarks like miniF2F (Zheng et al., 2022), FIMO (Liu et al., 2023), and PutnamBench (Tsoukalas et al., 2024), alongside advances in AI-assisted approaches such as AlphaGeometry (Trinh et al., 2024; Chervonyi et al., 2025) and recent neural theorem provers (Polu & Sutskever, 2020). Current verification relies primarily on interactive theorem provers like Lean4 (de Moura et al., 2015; de Moura & Ullrich, 2021), Coq (Huet & Paulin-Mohring, 2000), and Isabelle (Nipkow et al., 2002), with recent improvements in compilation efficiency through Kimina Lean Server (Santos et al., 2025). However, these approaches have critical limitations for RLVR applications: they only support formal languages while excluding natural language proofs, and provide only binary verification results without the detailed error analysis or actionable feedback essential for effective reinforcement learning.

Reward modeling for mathematical reasoning has primarily focused on traditional problem-solving tasks. Process reward models (Lightman et al., 2024; Wang et al., 2024a) have shown success in step-by-step verification for computational problems, while outcome reward models achieve strong results through final answer comparison (Cobbe et al., 2021). More broadly, LLM-as-a-judge approaches (Zheng et al., 2023) have evolved from instruction-following evaluation to reward models across domains including medical QA (Croxford et al., 2025), multimodal tasks (Chen et al., 2024), and code generation (Zhao et al., 2024), with open-source alternatives like PandaLM (Wang et al., 2024b) and Prometheus (Kim et al., 2024a;b) providing cost-effective solutions. While these methods succeed in traditional mathematical tasks through reference answer comparison (Chen et al., 2025), theorem proving's process-oriented nature and diverse solution paths present unique verification challenges that limit RLVR applications in this domain. Our work addresses this verification bottleneck by developing the first dual-language verifier specifically designed for mathematical theorem proving, enabling RLVR applications through detailed error analysis and fine-grained evaluation capabilities via novel two-stage reward modeling.

6 Conclusion

We introduce Proof-Verifier, a generative verifier that enables effective verification for mathematical theorem proofs in both formal and natural languages. Through a novel data synthesis pipeline and two-stage coarse-to-fine training framework, our model learns to provide quantitative assessment and detailed qualitative feedback for proof attempts. Experimental results show that Proof-Verifier achieves strong verification accuracy with high correlation to human judgment and ATP in model ranking. Our verifier effectively supports test-time scaling through best-of-n selection and enables iterative proof refinement via actionable feedback. Finally, we demonstrate that Proof-Verifier serves as an effective verifiable reward model for RLVR, revealing the potential of LLM-based verifiers to advance automated reasoning in domains without reference answers.

Ethics statement

This work proposes Proof-Verifier to enable Reinforcement Learning from Verifiable Rewards for mathematical theorem proving, enhancing models' mathematical reasoning capabilities. By generating detailed mathematical proof processes rather than potentially error-prone intermediate steps, our approach increases the trustworthiness of LLM outputs. This advancement strengthens the reliability and educational value of AI systems for mathematical learning and instruction, promoting more transparent and verifiable mathematical reasoning.

REPRODICIBILITY STATEMENT

To ensure reproducibility of our results, we provide comprehensive implementation details and experimental specifications throughout the paper and supplementary materials. Section 2 and Section 4 detail our training methodology, model architectures, and experimental setup, while the complete data processing pipeline, including our formal-to-informal translation procedure and filtering strategies, is described in Section 2.1 and Appendix F. All model configurations, hyperparameters, training procedures, and prompt templates used in our experiments are documented in Appendices G and M. To validate the quality of our human evaluation process, we include detailed examples of our annotation interface and inter-annotator agreement scores in Appendix D, along with sample cases demonstrating the assessment criteria used by human evaluators. The statistical distributions of our training datasets, experimental configurations for all baseline comparisons, and additional implementation details are provided in Appendices F through N. All materials necessary for reproducing our experimental results, including dataset construction procedures, evaluation protocols, and analysis methods, are included in the submitted appendices.

REFERENCES

Zhangir Azerbayev, Bartosz Piotrowski, Hailey Schoelkopf, Edward W. Ayers, Dragomir Radev, and Jeremy Avigad. Proofnet: Autoformalizing and formally proving undergraduate-level mathematics, 2023. URL https://arxiv.org/abs/2302.12433.

Ding Chen, Qingchen Yu, Pengyuan Wang, Wentao Zhang, Bo Tang, Feiyu Xiong, Xinchi Li, Minchuan Yang, and Zhiyu Li. xverify: Efficient answer verifier for reasoning model evaluations, 2025. URL https://arxiv.org/abs/2504.10481.

Dongping Chen, Ruoxi Chen, Shilin Zhang, Yinuo Liu, Yaochen Wang, Huichi Zhou, Qihui Zhang, Yao Wan, Pan Zhou, and Lichao Sun. Mllm-as-a-judge: Assessing multimodal llm-as-a-judge with vision-language benchmark, 2024. URL https://arxiv.org/abs/2402.04788.

Yuri Chervonyi, Trieu H. Trinh, Miroslav Olšák, Xiaomeng Yang, Hoang Nguyen, Marcelo Menegali, Junehyuk Jung, Vikas Verma, Quoc V. Le, and Thang Luong. Gold-medalist performance in solving olympiad geometry with alphageometry2, 2025. URL https://arxiv.org/abs/2502.03544.

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

Jacob Cohen. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46, 1960. doi: 10.1177/001316446002000104.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, Luke Marris, Sam Petulla, Colin Gaffney, Asaf Aharoni, Nathan Lintz, Tiago Cardal Pais, Henrik Jacobsson, Idan Szpektor, Nan-Jiang Jiang, Krishna Haridasan, Ahmed Omran, Nikunj Saunshi, Dara Bahri, Gaurav Mishra, Eric Chu, Toby Boyd, Brad Hekman, Aaron Parisi, Chaoyi Zhang, Kornraphop Kawintiranon, Tania Bedrax-Weiss, Oliver Wang, Ya Xu, Ollie Purkiss, Uri Mendlovic, Ilaï Deutel, Nam Nguyen, Adam Langley, Flip Korn, Lucia Rossazza, Alexandre Ramé, Sagar Waghmare, Helen Miller, Nathan Byrd, Ashrith Sheshan, Raia Hadsell Sangnie Bhardwaj, Pawel Janus, Tero Rissa, Dan Horgan, Sharon Silver, Ayzaan Wahid, Sergey Brin, Yves Raimond, Klemen Kloboves, Cindy Wang, Nitesh Bharadwaj Gundavarapu, Ilia Shumailov, Bo Wang, Mantas Pajarskas, Joe Heyward, Martin Nikoltchev, Maciej Kula, Hao Zhou, Zachary Garrett, Sushant Kafle, Sercan Arik, Ankita Goel, Mingyao Yang, Jiho Park, Koji Kojima, Parsa Mahmoudieh, Koray Kavukcuoglu, Grace Chen, Doug Fritz, Anton Bulyenov, Sudeshna Roy, Dimitris Paparas, Hadar Shemtov, Bo-Juen Chen, Robin Strudel, David Reitter, Aurko Roy, Andrey Vlasov, Changwan Ryu, Chas Leichner, Haichuan Yang, Zelda Mariet, Denis Vnukov, Tim Sohn, Amy Stuart, Wei Liang, Minmin Chen, Praynaa Rawlani, Christy Koh, JD Co-Reyes, Guangda Lai, Praseem Banzal, Dimitrios Vytiniotis, Jieru Mei, Mu Cai, Mohammed Badawi, Corey Fry, Ale Hartman, Daniel Zheng, Eric Jia, James Keeling, Annie Louis, Ying Chen, Efren Robles, Wei-Chih Hung, Howard Zhou, Nikita Saxena, Sonam Goenka, Olivia Ma, Zach Fisher, Mor Hazan Taege, Emily Graves, David Steiner, Yujia Li, Sarah Nguyen, Rahul Sukthankar, Joe Stanton, Ali Eslami, Gloria Shen, Berkin Akin, Alexey Guseynov, Yiqian Zhou, Jean-Baptiste Alayrac, Armand Joulin, Efrat Farkash, Ashish Thapliyal, Stephen Roller, Noam Shazeer, Todor Davchev, Terry Koo, Hannah Forbes-Pollard, Kartik Audhkhasi, Greg Farquhar, Adi Mayrav Gilady, Maggie Song, John Aslanides, Piermaria Mendolicchio, Alicia Parrish, John Blitzer, Pramod Gupta, Xiaoen Ju, Xiaochen Yang, Puranjay Datta, Andrea Tacchetti, Sanket Vaibhav Mehta, Gregory Dibb, Shubham Gupta, Federico Piccinini, Raia Hadsell, Sujee Rajayogam, Jiepu Jiang, Patrick Griffin, Patrik Sundberg, Jamie Hayes, Alexey Frolov, Tian Xie, Adam Zhang, Kingshuk Dasgupta, Uday Kalra, Lior Shani, Klaus Macherey, Tzu-Kuo Huang, Liam MacDermed, Karthik Duddu, Paulo Zacchello, Zi Yang, Jessica Lo, Kai Hui, Matej Kastelic, Derek Gasaway, Qijun Tan, Summer Yue, Pablo Barrio, John Wieting, Weel Yang, Andrew Nystrom, Solomon Demmessie, Anselm Levskaya, Fabio Viola, Chetan Tekur, Greg Billock, George Necula, Mandar Joshi, Rylan Schaeffer, Swachhand Lokhande, Christina Sorokin, Pradeep Shenoy, Mia Chen, Mark Collier, Hongji Li, Taylor Bos, Nevan Wichers, Sun Jae Lee, Angéline Pouget, Santhosh Thangaraj, Kyriakos Axiotis, Phil Crone, Rachel Sterneck, Nikolai Chinaev, Victoria Krakovna, Oleksandr Ferludin, Ian Gemp, Stephanie Winkler, Dan Goldberg, Ivan Korotkov, Kefan Xiao, Malika Mehrotra, Sandeep Mariserla, Vihari Piratla, Terry Thurk, Khiem Pham, Hongxu Ma, Alexandre Senges, Ravi Kumar, Clemens Meyer, Ellie Talius, Nuo Wang Pierse, Ballie Sandhu, Horia Toma, Kuo Lin, Swaroop Nath, Tom Stone, Dorsa Sadigh, Nikita Gupta, Arthur Guez, Avi Singh, Matt Thomas, Tom Duerig, Yuan Gong, Richard Tanburn, Lydia Lihui Zhang, Phuong Dao, Mohamed Hammad, Sirui Xie, Shruti Rijhwani, Ben Murdoch, Duhyeon Kim, Will Thompson, Heng-Tze Cheng, Daniel Sohn, Pablo Sprechmann, Qiantong Xu, Srinivas Tadepalli, Peter Young, Ye Zhang, Hansa Srinivasan, Miranda Aperghis, Aditya Ayyar, Hen Fitoussi, Ryan Burnell, David Madras, Mike Dusenberry, Xi Xiong, Tayo Oguntebi, Ben Albrecht, Jörg Bornschein, Jovana Mitrović, Mason Dimarco, Bhargav Kanagal Shamanna, Premal Shah, Eren Sezener, Shyam Upadhyay, Dave Lacey, Craig Schiff, Sebastien Baur, Sanjay Ganapathy, Eva Schnider, Mateo Wirth, Connor Schenck, Andrey Simanovsky, Yi-Xuan Tan, Philipp Fränken, Dennis Duan, Bharath Mankalale, Nikhil Dhawan, Kevin Sequeira, Zichuan Wei, Shivanker Goel, Caglar Unlu, Yukun Zhu, Haitian Sun, Ananth Balashankar, Kurt Shuster, Megh Umekar, Mahmoud Alnahlawi, Aäron van den Oord, Kelly Chen, Yuexiang Zhai, Zihang Dai, Kuang-Huei Lee, Eric Doi, Lukas Zilka, Rohith Vallu, Disha Shrivastava, Jason Lee, Hisham Husain, Honglei Zhuang, Vincent Cohen-Addad, Jarred Barber, James Atwood, Adam Sadovsky, Quentin Wellens, Steven Hand, Arunkumar Rajendran, Aybuke Turker, CJ Carey, Yuanzhong Xu, Hagen Soltau, Zefei Li, Xinying Song, Conglong Li, Iurii Kemaev, Sasha Brown, Andrea Burns, Viorica Patraucean, Piotr Stanczyk, Renga Aravamudhan, Mathieu Blondel, Hila Noga, Lorenzo Blanco, Will Song, Michael Isard, Mandar Sharma, Reid Hayes, Dalia El Badawy, Avery Lamp, Itay Laish, Olga Kozlova, Kelvin Chan, Sahil Singla, Srinivas Sunkara, Mayank Upadhyay, Chang Liu, Aijun Bai, Jarek Wilkiewicz, Martin Zlocha, Jeremiah Liu, Zhuowan Li, Haiguang Li, Omer Barak, Ganna Raboshchuk, Jiho Choi, Fangyu Liu, Erik Jue, Mohit Sharma, Andreea Marzoca, Robert Busa-Fekete, Anna Korsun, Andre Elisseeff, Zhe Shen, Sara Mc Carthy, Kay Lamerigts, Anahita Hosseini, Hanzhao Lin, Charlie Chen, Fan Yang, Kushal Chauhan, Mark Omernick, Dawei Jia, Karina Zainullina, Demis Hassabis, Danny Vainstein, Ehsan Amid, Xiang Zhou, Ronny Votel, Eszter Vértes, Xinjian Li, Zongwei Zhou, Angeliki Lazaridou, Brendan McMahan, Arjun Narayanan, Hubert Soyer, Sujoy Basu, Kayi Lee, Bryan Perozzi, Qin Cao, Leonard Berrada, Rahul Arya, Ke Chen, Katrina, Xu, Matthias Lochbrunner, Alex Hofer, Sahand Sharifzadeh, Renjie Wu, Sally Goldman, Pranjal Awasthi, Xuezhi Wang, Yan Wu, Claire Sha, Biao Zhang, Maciej Mikuła, Filippo Graziano, Siobhan Mcloughlin, Irene Giannoumis, Youhei Namiki, Chase Malik, Carey Radebaugh, Jamie Hall, Ramiro Leal-Cavazos, Jianmin Chen, Vikas Sindhwani, David Kao, David Greene, Jordan Griffith, Chris Welty, Ceslee Montgomery, Toshihiro Yoshino, Liangzhe Yuan, Noah Goodman, Assaf Hurwitz Michaely, Kevin Lee, KP Sawhney, Wei Chen, Zheng Zheng, Megan Shum, Nikolay Savinov, Etienne Pot, Alex Pak, Morteza Zadimoghaddam, Sijal Bhatnagar, Yoad Lewenberg, Blair Kutzman, Ji Liu, Lesley Katzen, Jeremy Selier, Josip Djolonga, Dmitry Lepikhin, Kelvin Xu, Jacky Liang, Jiewen Tan, Benoit Schillings, Muge Ersoy, Pete Blois, Bernd Bandemer, Abhimanyu Singh, Sergei Lebedev, Pankaj Joshi, Adam R. Brown, Evan Palmer, Shreya Pathak, Komal Jalan, Fedir Zubach, Shuba Lall, Randall Parker, Alok Gunjan, Sergey Rogulenko, Sumit Sanghai, Zhaoqi Leng, Zoltan Egyed, Shixin Li, Maria Ivanova, Kostas Andriopoulos, Jin Xie, Elan Rosenfeld, Auriel Wright, Ankur Sharma, Xinyang Geng, Yicheng Wang, Sam Kwei, Renke Pan, Yujing Zhang, Gabby Wang, Xi Liu, Chak Yeung, Elizabeth Cole, Aviv Rosenberg, Zhen Yang, Phil Chen, George Polovets, Pranav Nair, Rohun Saxena, Josh Smith, Shuo yiin Chang, Aroma Mahendru, Svetlana Grant, Anand Iyer, Irene Cai, Jed McGiffin, Jiaming Shen, Alanna Walton, Antonious Girgis, Oliver Woodman, Rosemary Ke, Mike Kwong, Louis Rouillard, Jinmeng Rao, Zhihao Li, Yuntao Xu, Flavien Prost, Chi Zou, Ziwei Ji, Alberto Magni, Tyler Liechty, Dan A. Calian, Deepak Ramachandran, Igor Krivokon, Hui Huang, Terry Chen, Anja Hauth, Anastasija Ilić, Weijuan Xi, Hyeontaek Lim, Vlad-Doru Ion, Pooya Moradi, Metin Toksoz-Exley, Kalesha Bullard, Miltos Allamanis, Xiaomeng Yang, Sophie Wang, Zhi Hong, Anita Gergely, Cheng Li, Bhavishya Mittal, Vitaly Kovalev, Victor Ungureanu, Jane Labanowski, Jan Wassenberg, Nicolas Lacasse, Geoffrey Cideron, Petar Dević, Annie Marsden, Lynn Nguyen, Michael Fink, Yin Zhong, Tatsuya Kiyono, Desi Ivanov, Sally Ma, Max Bain, Kiran Yalasangi, Jennifer She, Anastasia Petrushkina, Mayank Lunayach, Carla Bromberg, Sarah Hodkinson, Vilobh Meshram, Daniel Vlasic, Austin Kyker, Steve Xu, Jeff Stanway, Zuguang Yang, Kai Zhao, Matthew Tung, Seth Odoom, Yasuhisa Fujii, Justin Gilmer, Eunyoung Kim, Felix Halim, Quoc Le, Bernd Bohnet, Seliem El-Sayed, Behnam Neyshabur, Malcolm Reynolds, Dean Reich, Yang Xu, Erica Moreira, Anuj Sharma, Zeyu Liu, Mohammad Javad Hosseini, Naina Raisinghani, Yi Su, Ni Lao, Daniel Formoso, Marco Gelmi, Almog Gueta, Tapomay Dey, Elena Gribovskaya, Domagoj Ćevid, Sidharth Mudgal, Garrett Bingham, Jianling Wang, Anurag Kumar, Alex Cullum, Feng Han, Konstantinos Bousmalis, Diego Cedillo, Grace Chu, Vladimir Magay, Paul Michel, Ester Hlavnova, Daniele Calandriello, Setareh Ariafar, Kaisheng Yao, Vikash Sehwag, Arpi Vezer, Agustin Dal Lago, Zhenkai Zhu, Paul Kishan Rubenstein, Allen Porter, Anirudh Baddepudi, Oriana Riva, Mihai Dorin Istin, Chih-Kuan Yeh, Zhi Li, Andrew Howard, Nilpa Jha, Jeremy Chen, Raoul de Liedekerke, Zafarali Ahmed, Mikel Rodriguez, Tanuj Bhatia, Bangju Wang, Ali Elqursh, David Klinghoffer, Peter Chen, Pushmeet Kohli, Te I, Weiyang Zhang, Zack Nado, Jilin Chen, Maxwell Chen, George Zhang, Aayush Singh, Adam Hillier, Federico Lebron, Yiqing Tao, Ting Liu, Gabriel Dulac-Arnold, Jingwei Zhang, Shashi Narayan, Buhuang Liu, Orhan Firat, Abhishek Bhowmick, Bingyuan Liu, Hao Zhang, Zizhao Zhang, Georges Rotival, Nathan Howard, Anu Sinha, Alexander Grushetsky, Benjamin Beyret, Keerthana Gopalakrishnan, James Zhao, Kyle He, Szabolcs Payrits, Zaid Nabulsi, Zhaoyi Zhang, Weijie Chen, Edward Lee, Nova Fallen, Sreenivas Gollapudi, Aurick Zhou, Filip Pavetić, Thomas Köppe, Shiyu Huang, Rama Pasumarthi, Nick Fernando, Felix Fischer, Daria Ćurko, Yang Gao, James Svensson, Austin Stone, Haroon Qureshi, Abhishek Sinha, Apoorv Kulshreshtha, Martin Matysiak, Jieming Mao, Carl Saroufim, Aleksandra Faust, Qingnan Duan, Gil Fidel, Kaan Katircioglu, Raphaël Lopez Kaufman, Dhruv Shah, Weize Kong, Abhishek Bapna, Gellért Weisz, Emma Dunleavy, Praneet Dutta, Tianqi Liu, Rahma Chaabouni, Carolina Parada, Marcus Wu, Alexandra Belias, Alessandro Bissacco, Stanislav Fort, Li Xiao, Fantine Huot, Chris Knutsen, Yochai Blau, Gang Li, Jennifer Prendki, Juliette Love, Yinlam Chow, Pichi Charoenpanit, Hidetoshi Shimokawa, Vincent Coriou, Karol Gregor, Tomas Izo, Arjun Akula, Mario Pinto, Chris Hahn, Dominik Paulus, Jiaxian Guo, Neha Sharma, Cho-Jui Hsieh, Adaeze Chukwuka, Kazuma Hashimoto, Nathalie Rauschmayr, Ling Wu, Christof Angermueller, Yulong Wang, Sebastian Gerlach, Michael Pliskin, Daniil Mirylenka, Min Ma, Lexi Baugher, Bryan Gale, Shaan Bijwadia, Nemanja Rakićević, David Wood, Jane Park, Chung-Ching Chang, Babi Seal, Chris Tar, Kacper Krasowiak, Yiwen Song, Georgi Stephanov, Gary Wang, Marcello Maggioni, Stein Xudong Lin, Felix Wu, Shachi Paul, Zixuan Jiang, Shubham Agrawal, Bilal Piot, Alex Feng, Cheolmin Kim, Tulsee Doshi, Jonathan Lai, Chuqiao, Xu, Sharad Vikram, Ciprian Chelba, Sebastian Krause, Vincent Zhuang, Jack Rae, Timo Denk, Adrian Collister, Lotte Weerts, Xianghong Luo, Yifeng Lu, Håvard Garnes, Nitish Gupta, Terry Spitz, Avinatan Hassidim, Lihao Liang, Izhak Shafran, Peter Humphreys, Kenny Vassigh, Phil Wallis, Virat Shejwalkar, Nicolas Perez-Nieves, Rachel Hornung, Melissa Tan, Beka Westberg, Andy Ly, Richard Zhang, Brian Farris, Jongbin Park, Alec Kosik, Zeynep Cankara, Andrii Maksai, Yunhan Xu, Albin Cassirer, Sergi Caelles, Abbas Abdolmaleki, Mencher Chiang, Alex Fabrikant, Shravya Shetty, Luheng He, Mai Giménez, Hadi Hashemi, Sheena Panthaplackel, Yana Kulizhskaya, Salil Deshmukh, Daniele Pighin, Robin Alazard, Disha Jindal, Seb Noury, Pradeep Kumar S, Siyang Qin, Xerxes Dotiwalla, Stephen Spencer, Mohammad Babaeizadeh, Blake JianHang Chen, Vaibhav Mehta, Jennie Lees, Andrew Leach, Penporn Koanantakool, Ilia Akolzin, Ramona Comanescu, Junwhan Ahn, Alexey Svyatkovskiy, Basil Mustafa, David D'Ambrosio, Shiva Mohan Reddy Garlapati, Pascal Lamblin, Alekh Agarwal, Shuang Song, Pier Giuseppe Sessa, Pauline Coquinot, John Maggs, Hussain Masoom, Divya Pitta, Yaqing Wang, Patrick Morris-Suzuki, Billy Porter, Johnson Jia, Jeffrey Dudek, Raghavender R, Cosmin Paduraru, Alan Ansell, Tolga Bolukbasi, Tony Lu, Ramya Ganeshan, Zi Wang, Henry Griffiths, Rodrigo Benenson, Yifan He, James Swirhun, George Papamakarios, Aditya Chawla, Kuntal Sengupta, Yan Wang, Vedrana Milutinovic, Igor Mordatch, Zhipeng Jia, Jamie Smith, Will Ng, Shitij Nigam, Matt Young, Eugen Vušak, Blake Hechtman, Sheela Goenka, Avital Zipori, Kareem Ayoub, Ashok Popat, Trilok Acharya, Luo Yu, Dawn Bloxwich, Hugo Song, Paul Roit, Haiqiong Li, Aviel Boag, Nigamaa Nayakanti, Bilva Chandra, Tianli Ding, Aahil Mehta, Cath Hope, Jiageng Zhang, Idan Heimlich Shtacher, Kartikeya Badola, Ryo Nakashima, Andrei Sozanschi, Iulia Comşa, Ante Žužul, Emily Caveness, Julian Odell, Matthew Watson, Dario de Cesare, Phillip Lippe, Derek Lockhart, Siddharth Verma, Huizhong Chen, Sean Sun, Lin Zhuo, Aditya Shah, Prakhar Gupta, Alex Muzio, Ning Niu, Amir Zait, Abhinav Singh, Meenu Gaba, Fan Ye, Prajit Ramachandran, Mohammad Saleh, Raluca Ada Popa, Ayush Dubey, Frederick Liu, Sara Javanmardi, Mark Epstein, Ross Hemsley, Richard Green, Nishant Ranka, Eden Cohen, Chuyuan Kelly Fu, Sanjay Ghemawat, Jed Borovik, James Martens, Anthony Chen, Pranav Shyam, André Susano Pinto, Ming-Hsuan Yang, Alexandru Ţifrea, David Du, Boqing Gong, Ayushi Agarwal, Seungyeon Kim, Christian Frank, Saloni Shah, Xiaodan Song, Zhiwei Deng, Ales Mikhalap, Kleopatra Chatziprimou, Timothy Chung, Toni Creswell, Susan Zhang, Yennie Jun, Carl Lebsack, Will Truong, Slavica Andačić, Itay Yona, Marco Fornoni, Rong Rong, Serge Toropov, Afzal Shama Soudagar, Andrew Audibert, Salah Zaiem, Zaheer Abbas, Andrei Rusu, Sahitya Potluri, Shitao Weng, Anastasios Kementsietsidis, Anton Tsitsulin, Daiyi Peng, Natalie Ha, Sanil Jain, Tejasi Latkar, Simeon Ivanov, Cory McLean, Anirudh GP, Rajesh Venkataraman, Canoee Liu, Dilip Krishnan, Joel D'sa, Roey Yogev, Paul Collins, Benjamin Lee, Lewis Ho, Carl Doersch, Gal Yona, Shawn Gao, Felipe Tiengo Ferreira, Adnan Ozturel, Hannah Muckenhirn, Ce Zheng, Gargi Balasubramaniam, Mudit Bansal, George van den Driessche, Sivan Eiger, Salem Haykal, Vedant Misra, Abhimanyu Goyal, Danilo Martins, Gary Leung, Jonas Valfridsson, Four Flynn, Will Bishop, Chenxi Pang, Yoni Halpern, Honglin Yu, Lawrence Moore, Yuvein, Zhu, Sridhar Thiagarajan, Yoel Drori, Zhisheng Xiao, Lucio Dery, Rolf Jagerman, Jing Lu, Eric Ge, Vaibhav Aggarwal, Arjun Khare, Vinh Tran, Oded Elyada, Ferran Alet, James Rubin, Ian Chou, David Tian, Libin Bai, Lawrence Chan, Lukasz Lew, Karolis Misiunas, Taylan Bilal, Aniket Ray, Sindhu Raghuram, Alex Castro-Ros, Viral Carpenter, CJ Zheng, Michael Kilgore, Josef Broder, Emily Xue, Praveen Kallakuri, Dheeru Dua, Nancy Yuen, Steve Chien, John Schultz, Saurabh Agrawal, Reut Tsarfaty, Jingcao Hu, Ajay Kannan, Dror Marcus, Nisarg Kothari, Baochen Sun, Ben Horn, Matko Bošnjak, Ferjad Naeem, Dean Hirsch, Lewis Chiang, Boya Fang, Jie Han, Qifei Wang, Ben Hora, Antoine He, Mario Lučić, Beer Changpinyo, Anshuman Tripathi, John Youssef, Chester Kwak, Philippe Schlattner, Cat Graves, Rémi Leblond, Wenjun Zeng, Anders Andreassen, Gabriel Rasskin, Yue Song, Eddie Cao, Junhyuk Oh, Matt Hoffman, Wojtek Skut, Yichi Zhang, Jon Stritar, Xingyu Cai, Saarthak Khanna, Kathie Wang, Shriya Sharma, Christian Reisswig, Younghoon Jun, Aman Prasad, Tatiana Sholokhova, Preeti Singh, Adi Gerzi Rosenthal, Anian Ruoss, Françoise Beaufays, Sean Kirmani, Dongkai Chen, Johan Schalkwyk, Jonathan Herzig, Been Kim, Josh Jacob, Damien Vincent, Adrian N Reyes, Ivana Balazevic, Léonard Hussenot, Jon Schneider, Parker Barnes, Luis Castro, Spandana Raj Babbula, Simon Green, Serkan Cabi, Nico Duduta, Danny Driess, Rich Galt, Noam Velan, Junjie Wang, Hongyang Jiao, Matthew Mauger, Du Phan, Miteyan Patel, Vlado Galić, Jerry Chang, Eyal Marcus, Matt Harvey, Julian Salazar, Elahe Dabir, Suraj Satishkumar Sheth, Amol Mandhane, Hanie Sedghi, Jeremiah Willcock, Amir Zandieh, Shruthi Prabhakara, Aida Amini, Antoine Miech, Victor Stone, Massimo Nicosia, Paul Niemczyk, Ying Xiao, Lucy Kim, Sławek Kwasiborski, Vikas Verma, Ada Maksutaj Oflazer, Christoph Hirnschall, Peter Sung, Lu Liu, Richard Everett, Michiel Bakker, Ágoston Weisz, Yufei Wang, Vivek Sampathkumar, Uri Shaham, Bibo Xu, Yasemin Altun, Mingqiu Wang, Takaaki Saeki, Guanjie Chen, Emanuel Taropa, Shanthal Vasanth, Sophia Austin, Lu Huang, Goran Petrovic, Qingyun Dou, Daniel Golovin, Grigory Rozhdestvenskiy, Allie Culp, Will Wu, Motoki Sano, Divya Jain, Julia Proskurnia, Sébastien Cevey, Alejandro Cruzado Ruiz, Piyush Patil, Mahdi Mirzazadeh, Eric Ni, Javier Snaider, Lijie Fan, Alexandre Fréchette, AJ Pierigiovanni, Shariq Iqbal, Kenton Lee, Claudio Fantacci, Jinwei Xing, Lisa Wang, Alex Irpan, David Raposo, Yi Luan, Zhuoyuan Chen, Harish Ganapathy, Kevin Hui, Jiazhong Nie, Isabelle Guyon, Heming Ge, Roopali Vij, Hui Zheng, Dayeong Lee, Alfonso Castaño, Khuslen Baatarsukh, Gabriel Ibagon, Alexandra Chronopoulou, Nicholas FitzGerald, Shashank Viswanadha, Safeen Huda, Rivka Moroshko, Georgi Stoyanov, Prateek Kolhar, Alain Vaucher, Ishaan Watts, Adhi Kuncoro, Henryk Michalewski, Satish Kambala, Bat-Orgil Batsaikhan, Alek Andreev, Irina Jurenka, Maigo Le, Qihang Chen, Wael Al Jishi, Sarah Chakera, Zhe Chen, Aditya Kini, Vikas Yadav, Aditya Siddhant, Ilia Labzovsky, Balaji Lakshminarayanan, Carrie Grimes Bostock, Pankil Botadra, Ankesh Anand, Colton Bishop, Sam Conway-Rahman, Mohit Agarwal, Yani Donchev, Achintya Singhal, Félix de Chaumont Quitry, Natalia Ponomareva, Nishant Agrawal, Bin Ni, Kalpesh Krishna, Masha Samsikova, John Karro, Yilun Du, Tamara von Glehn, Caden Lu, Christopher A. Choquette-Choo, Zhen Qin, Tingnan Zhang, Sicheng Li, Divya Tyam, Swaroop Mishra, Wing Lowe, Colin Ji, Weiyi Wang, Manaal Faruqui, Ambrose Slone, Valentin Dalibard, Arunachalam Narayanaswamy, John Lambert, Pierre-Antoine Manzagol, Dan Karliner, Andrew Bolt, Ivan Lobov, Aditya Kusupati, Chang Ye, Xuan Yang, Heiga Zen, Nelson George, Mukul Bhutani, Olivier Lacombe, Robert Riachi, Gagan Bansal, Rachel Soh, Yue Gao, Yang Yu, Adams Yu, Emily Nottage, Tania Rojas-Esponda, James Noraky, Manish Gupta, Ragha Kotikalapudi, Jichuan Chang, Sanja Deur, Dan Graur, Alex Mossin, Erin Farnese, Ricardo Figueira, Alexandre Moufarek, Austin Huang, Patrik Zochbauer, Ben Ingram, Tongzhou Chen, Zelin Wu, Adrià Puigdomènech, Leland Rechis, Da Yu, Sri Gayatri Sundara Padmanabhan, Rui Zhu, Chu ling Ko, Andrea Banino, Samira Daruki, Aarush Selvan, Dhruva Bhaswar, Daniel Hernandez Diaz, Chen Su, Salvatore Scellato, Jennifer Brennan, Woohyun Han, Grace Chung, Priyanka Agrawal, Urvashi Khandelwal, Khe Chai Sim, Morgane Lustman, Sam Ritter, Kelvin Guu, Jiawei Xia, Prateek Jain, Emma Wang, Tyrone Hill, Mirko Rossini, Marija Kostelac, Tautvydas Misiunas, Amit Sabne, Kyuyeun Kim, Ahmet Iscen, Congchao Wang, José Leal, Ashwin Sreevatsa, Utku Evci, Manfred Warmuth, Saket Joshi, Daniel Suo, James Lottes, Garrett Honke, Brendan Jou, Stefani Karp, Jieru Hu, Himanshu Sahni, Adrien Ali Taïga, William Kong, Samrat Ghosh, Renshen Wang, Jay Pavagadhi, Natalie Axelsson, Nikolai Grigorev, Patrick Siegler, Rebecca Lin, Guohui Wang, Emilio Parisotto, Sharath Maddineni, Krishan Subudhi, Eyal Ben-David, Elena Pochernina, Orgad Keller, Thi Avrahami, Zhe Yuan, Pulkit Mehta, Jialu Liu, Sherry Yang, Wendy Kan, Katherine Lee, Tom Funkhouser, Derek Cheng, Hongzhi Shi, Archit Sharma, Joe Kelley, Matan Eyal, Yury Malkov, Corentin Tallec, Yuval Bahat, Shen Yan, Xintian, Wu, David Lindner, Chengda Wu, Avi Caciularu, Xiyang Luo, Rodolphe Jenatton, Tim Zaman, Yingying Bi, Ilya Kornakov, Ganesh Mallya, Daisuke Ikeda, Itay Karo, Anima Singh, Colin Evans, Praneeth Netrapalli, Vincent Nallatamby, Isaac Tian, Yannis Assael, Vikas Raunak, Victor Carbune, Ioana Bica, Lior Madmoni, Dee Cattle, Snchit Grover, Krishna Somandepalli, Sid Lall, Amelio Vázquez-Reina, Riccardo Patana, Jiaqi Mu, Pranav Talluri, Maggie Tran, Rajeev Aggarwal, RJ Skerry-Ryan, Jun Xu, Mike Burrows, Xiaoyue Pan, Edouard Yvinec, Di Lu, Zhiying Zhang, Duc Dung Nguyen, Hairong Mu, Gabriel Barcik, Helen Ran, Lauren Beltrone, Krzysztof Choromanski, Dia Kharrat, Samuel Albanie, Sean Purser-haskell, David Bieber, Carrie Zhang, Jing Wang, Tom Hudson, Zhiyuan Zhang, Han Fu, Johannes Mauerer, Mohammad Hossein Bateni, AJ Maschinot, Bing Wang, Muye Zhu, Arjun Pillai, Tobias Weyand, Shuang Liu, Oscar Akerlund, Fred Bertsch, Vittal Premachandran, Alicia Jin, Vincent Roulet, Peter de Boursac, Shubham Mittal, Ndaba Ndebele, Georgi Karadzhov, Sahra Ghalebikesabi, Ricky Liang, Allen Wu, Yale Cong, Nimesh Ghelani, Sumeet Singh, Bahar Fatemi, Warren, Chen, Charles Kwong, Alexey Kolganov, Steve Li, Richard Song, Chenkai Kuang, Sobhan Miryoosefi, Dale Webster, James Wendt, Arkadiusz Socala, Guolong Su, Artur Mendonça, Abhinav Gupta, Xiaowei Li, Tomy Tsai, Qiong, Hu, Kai Kang, Angie Chen, Sertan Girgin, Yongqin Xian, Andrew Lee, Nolan Ramsden, Leslie Baker, Madeleine Clare Elish, Varvara Krayvanova, Rishabh Joshi, Jiri Simsa, Yao-Yuan Yang, Piotr Ambroszczyk, Dipankar Ghosh, Arjun Kar, Yuan Shangguan, Yumeya Yamamori, Yaroslav Akulov, Andy Brock, Haotian Tang, Siddharth Vashishtha, Rich Munoz, Andreas Steiner, Kalyan Andra, Daniel Eppens, Qixuan Feng, Hayato Kobayashi, Sasha Goldshtein, Mona El Mahdy, Xin Wang, Jilei, Wang, Richard Killam, Tom Kwiatkowski, Kavya Kopparapu, Serena Zhan, Chao Jia, Alexei Bendebury, Sheryl Luo, Adrià Recasens, Timothy Knight, Jing Chen, Mohak Patel, YaGuang Li, Ben Withbroe, Dean Weesner, Kush Bhatia, Jie Ren, Danielle Eisenbud, Ebrahim Songhori, Yanhua Sun, Travis Choma, Tasos Kementsietsidis, Lucas Manning, Brian Roark, Wael Farhan, Jie Feng, Susheel Tatineni, James Cobon-Kerr, Yunjie Li, Lisa Anne Hendricks, Isaac Noble, Chris Breaux, Nate Kushman, Liqian Peng, Fuzhao Xue, Taylor Tobin, Jamie Rogers, Josh Lipschultz, Chris Alberti, Alexey Vlaskin, Mostafa Dehghani, Roshan Sharma, Tris Warkentin, Chen-Yu Lee, Benigno Uria, Da-Cheng Juan, Angad Chandorkar, Hila Sheftel, Ruibo Liu, Elnaz Davoodi, Borja De Balle Pigem, Kedar Dhamdhere, David Ross, Jonathan Hoech, Mahdis Mahdieh, Li Liu, Qiujia Li, Liam McCafferty, Chenxi Liu, Markus Mircea, Yunting Song, Omkar Savant, Alaa Saade, Colin Cherry, Vincent Hellendoorn, Siddharth Goyal, Paul Pucciarelli, David Vilar Torres, Zohar Yahav, Hyo Lee, Lars Lowe Sjoesund, Christo Kirov, Bo Chang, Deepanway Ghoshal, Lu Li, Gilles Baechler, Sébastien Pereira, Tara Sainath, Anudhyan Boral, Dominik Grewe, Afief Halumi, Nguyet Minh Phu, Tianxiao Shen, Marco Tulio Ribeiro, Dhriti Varma, Alex Kaskasoli, Vlad Feinberg, Navneet Potti, Jarrod Kahn, Matheus Wisniewski, Shakir Mohamed, Arnar Mar Hrafnkelsson, Bobak Shahriari, Jean-Baptiste Lespiau, Lisa Patel, Legg Yeung, Tom Paine, Lantao Mei, Alex Ramirez, Rakesh Shivanna, Li Zhong, Josh Woodward, Guilherme Tubone, Samira Khan, Heng Chen, Elizabeth Nielsen, Catalin Ionescu, Utsav Prabhu, Mingcen Gao, Qingze Wang, Sean Augenstein, Neesha Subramaniam, Jason Chang, Fotis Iliopoulos, Jiaming Luo, Myriam Khan, Weicheng Kuo, Denis Teplyashin, Florence Perot, Logan Kilpatrick, Amir Globerson, Hongkun Yu, Anfal Siddiqui, Nick Sukhanov, Arun Kandoor, Umang Gupta, Marco Andreetto, Moran Ambar, Donnie Kim, Paweł Wesołowski, Sarah Perrin, Ben Limonchik, Wei Fan, Jim Stephan, Ian Stewart-Binks, Ryan Kappedal, Tong He, Sarah Cogan, Romina Datta, Tong Zhou, Jiayu Ye, Leandro Kieliger, Ana Ramalho, Kyle Kastner, Fabian Mentzer, Wei-Jen Ko, Arun Suggala, Tianhao Zhou, Shiraz Butt, Hana Strejček, Lior Belenki, Subhashini Venugopalan, Mingyang Ling, Evgenii Eltyshev, Yunxiao Deng, Geza Kovacs, Mukund Raghavachari, Hanjun Dai, Tal Schuster, Steven Schwarcz, Richard Nguyen, Arthur Nguyen, Gavin Buttimore, Shrestha Basu Mallick, Sudeep Gandhe, Seth Benjamin, Michal Jastrzebski, Le Yan, Sugato Basu, Chris Apps, Isabel Edkins, James Allingham, Immanuel Odisho, Tomas Kocisky, Jewel Zhao, Linting Xue, Apoorv Reddy, Chrysovalantis Anastasiou, Aviel Atias, Sam Redmond, Kieran Milan, Nicolas Heess, Herman Schmit, Allan Dafoe, Daniel Andor, Tynan Gangwani, Anca Dragan, Sheng Zhang, Ashyana Kachra, Gang Wu, Siyang Xue, Kevin Aydin, Siqi Liu, Yuxiang Zhou, Mahan Malihi, Austin Wu, Siddharth Gopal, Candice Schumann, Peter Stys, Alek Wang, Mirek Olšák, Dangyi Liu, Christian Schallhart, Yiran Mao, Demetra Brady, Hao Xu, Tomas Mery, Chawin Sitawarin, Siva Velusamy, Tom Cobley, Alex Zhai, Christian Walder, Nitzan Katz, Ganesh Jawahar, Chinmay Kulkarni, Antoine Yang, Adam Paszke, Yinan Wang, Bogdan Damoc, Zalán Borsos, Ray Smith, Jinning Li, Mansi Gupta, Andrei Kapishnikov, Sushant Prakash, Florian Luisier, Rishabh Agarwal, Will Grathwohl, Kuangyuan Chen, Kehang Han, Nikhil Mehta, Andrew Over, Shekoofeh Azizi, Lei Meng, Niccolò Dal Santo, Kelvin Zheng, Jane Shapiro, Igor Petrovski, Jeffrey Hui, Amin Ghafouri, Jasper Snoek, James Qin, Mandy Jordan, Caitlin Sikora, Jonathan Malmaud, Yuheng Kuang, Aga Świetlik, Ruoxin Sang, Chongyang Shi, Leon Li, Andrew Rosenberg, Shubin Zhao, Andy Crawford, Jan-Thorsten Peter, Yun Lei, Xavier Garcia, Long Le, Todd Wang, Julien Amelot, Dave Orr, Praneeth Kacham, Dana Alon, Gladys Tyen, Abhinav Arora, James Lyon, Alex Kurakin, Mimi Ly, Theo Guidroz, Zhipeng Yan, Rina Panigrahy, Pingmei Xu, Thais Kagohara, Yong Cheng, Eric Noland, Jinhyuk Lee, Jonathan Lee, Cathy Yip, Maria Wang, Efrat Nehoran, Alexander Bykovsky, Zhihao Shan, Ankit Bhagatwala, Chaochao Yan, Jie Tan, Guillermo Garrido, Dan Ethier, Nate Hurley, Grace Vesom, Xu Chen, Siyuan Qiao, Abhishek Nayyar, Julian Walker, Paramjit Sandhu, Mihaela Rosca, Danny Swisher, Mikhail Dektiarev, Josh Dillon, George-Cristian Muraru, Manuel Tragut, Artiom Myaskovsky, David Reid, Marko Velic, Owen Xiao, Jasmine George, Mark Brand, Jing Li, Wenhao Yu, Shane Gu, Xiang Deng, François-Xavier Aubet, Soheil Hassas Yeganeh, Fred Alcober, Celine Smith, Trevor Cohn, Kay McKinney, Michael Tschannen, Ramesh Sampath, Gowoon Cheon, Liangchen Luo, Luyang Liu, Jordi Orbay, Hui Peng, Gabriela Botea, Xiaofan Zhang, Charles Yoon, Cesar Magalhaes, Paweł Stradomski, Ian Mackinnon, Steven Hemingray, Kumaran Venkatesan, Rhys May, Jaeyoun Kim, Alex Druinsky, Jingchen Ye, Zheng Xu, Terry Huang, Jad Al Abdallah, Adil Dostmohamed, Rachana Fellinger, Tsendsuren Munkhdalai, Akanksha Maurya, Peter Garst, Yin Zhang, Maxim Krikun, Simon Bucher, Aditya Srikanth Veerubhotla, Yaxin Liu, Sheng Li, Nishesh Gupta, Jakub Adamek, Hanwen Chen, Bernett Orlando, Aleksandr Zaks, Joost van Amersfoort, Josh Camp, Hui Wan, HyunJeong Choe, Zhichun Wu, Kate Olszewska, Weiren Yu, Archita Vadali, Martin Scholz, Daniel De Freitas, Jason Lin, Amy Hua, Xin Liu, Frank Ding, Yichao Zhou, Boone Severson, Katerina Tsihlas, Samuel Yang, Tammo Spalink, Varun Yerram, Helena Pankov, Rory Blevins, Ben Vargas, Sarthak Jauhari, Matt Miecnikowski, Ming Zhang, Sandeep Kumar, Clement Farabet, Charline Le Lan, Sebastian Flennerhag, Yonatan Bitton, Ada Ma, Arthur Bražinskas, Eli Collins, Niharika Ahuja, Sneha Kudugunta, Anna Bortsova, Minh Giang, Wanzheng Zhu, Ed Chi, Scott Lundberg, Alexey Stern, Subha Puttagunta, Jing Xiong, Xiao Wu, Yash Pande, Amit Jhindal, Daniel Murphy, Jon Clark, Marc Brockschmidt, Maxine Deines, Kevin R. McKee, Dan Bahir, Jiajun Shen, Minh Truong, Daniel McDuff, Andrea Gesmundo, Edouard Rosseel, Bowen Liang, Ken Caluwaerts, Jessica Hamrick, Joseph Kready, Mary Cassin, Rishikesh Ingale, Li Lao, Scott Pollom, Yifan Ding, Wei He, Lizzetth Bellot, Joana Iljazi, Ramya Sree Boppana, Shan Han, Tara

Thompson, Amr Khalifa, Anna Bulanova, Blagoj Mitrevski, Bo Pang, Emma Cooney, Tian Shi, Rey Coaguila, Tamar Yakar, Marc'aurelio Ranzato, Nikola Momchev, Chris Rawles, Zachary Charles, Young Maeng, Yuan Zhang, Rishabh Bansal, Xiaokai Zhao, Brian Albert, Yuan Yuan, Sudheendra Vijayanarasimhan, Roy Hirsch, Vinay Ramasesh, Kiran Vodrahalli, Xingyu Wang, Arushi Gupta, DJ Strouse, Jianmo Ni, Roma Patel, Gabe Taubman, Zhouyuan Huo, Dero Gharibian, Marianne Monteiro, Hoi Lam, Shobha Vasudevan, Aditi Chaudhary, Isabela Albuquerque, Kilol Gupta, Sebastian Riedel, Chaitra Hegde, Avraham Ruderman, András György, Marcus Wainwright, Ashwin Chaugule, Burcu Karagol Ayan, Tomer Levinboim, Sam Shleifer, Yogesh Kalley, Vahab Mirrokni, Abhishek Rao, Prabakar Radhakrishnan, Jay Hartford, Jialin Wu, Zhenhai Zhu, Francesco Bertolini, Hao Xiong, Nicolas Serrano, Hamish Tomlinson, Myle Ott, Yifan Chang, Mark Graham, Jian Li, Marco Liang, Xiangzhu Long, Sebastian Borgeaud, Yanif Ahmad, Alex Grills, Diana Mincu, Martin Izzard, Yuan Liu, Jinyu Xie, Louis O'Bryan, Sameera Ponda, Simon Tong, Michelle Liu, Dan Malkin, Khalid Salama, Yuankai Chen, Rohan Anil, Anand Rao, Rigel Swavely, Misha Bilenko, Nina Anderson, Tat Tan, Jing Xie, Xing Wu, Lijun Yu, Oriol Vinyals, Andrey Ryabtsev, Rumen Dangovski, Kate Baumli, Daniel Keysers, Christian Wright, Zoe Ashwood, Betty Chan, Artem Shtefan, Yaohui Guo, Ankur Bapna, Radu Soricut, Steven Pecht, Sabela Ramos, Rui Wang, Jiahao Cai, Trieu Trinh, Paul Barham, Linda Friso, Eli Stickgold, Xiangzhuo Ding, Siamak Shakeri, Diego Ardila, Eleftheria Briakou, Phil Culliton, Adam Raveret, Jingyu Cui, David Saxton, Subhrajit Roy, Javad Azizi, Pengcheng Yin, Lucia Loher, Andrew Bunner, Min Choi, Faruk Ahmed, Eric Li, Yin Li, Shengyang Dai, Michael Elabd, Sriram Ganapathy, Shivani Agrawal, Yiqing Hua, Paige Kunkle, Sujeevan Rajayogam, Arun Ahuja, Arthur Conmy, Alex Vasiloff, Parker Beak, Christopher Yew, Jayaram Mudigonda, Bartek Wydrowski, Jon Blanton, Zhengdong Wang, Yann Dauphin, Zhuo Xu, Martin Polacek, Xi Chen, Hexiang Hu, Pauline Sho, Markus Kunesch, Mehdi Hafezi Manshadi, Eliza Rutherford, Bo Li, Sissie Hsiao, Iain Barr, Alex Tudor, Matija Kecman, Arsha Nagrani, Vladimir Pchelin, Martin Sundermeyer, Aishwarya P S, Abhijit Karmarkar, Yi Gao, Grishma Chole, Olivier Bachem, Isabel Gao, Arturo BC, Matt Dibb, Mauro Verzetti, Felix Hernandez-Campos, Yana Lunts, Matthew Johnson, Julia Di Trapani, Raphael Koster, Idan Brusilovsky, Binbin Xiong, Megha Mohabey, Han Ke, Joe Zou, Tea Sabolić, Víctor Campos, John Palowitch, Alex Morris, Linhai Qiu, Pranavaraj Ponnuramu, Fangtao Li, Vivek Sharma, Kiranbir Sodhia, Kaan Tekelioglu, Aleksandr Chuklin, Madhavi Yenugula, Erika Gemzer, Theofilos Strinopoulos, Sam El-Husseini, Huiyu Wang, Yan Zhong, Edouard Leurent, Paul Natsev, Weijun Wang, Dre Mahaarachchi, Tao Zhu, Songyou Peng, Sami Alabed, Cheng-Chun Lee, Anthony Brohan, Arthur Szlam, GS Oh, Anton Kovsharov, Jenny Lee, Renee Wong, Megan Barnes, Gregory Thornton, Felix Gimeno, Omer Levy, Martin Sevenich, Melvin Johnson, Jonathan Mallinson, Robert Dadashi, Ziyue Wang, Qingchun Ren, Preethi Lahoti, Arka Dhar, Josh Feldman, Dan Zheng, Thatcher Ulrich, Liviu Panait, Michiel Blokzijl, Cip Baetu, Josip Matak, Jitendra Harlalka, Maulik Shah, Tal Marian, Daniel von Dincklage, Cosmo Du, Ruy Ley-Wild, Bethanie Brownfield, Max Schumacher, Yury Stuken, Shadi Noghabi, Sonal Gupta, Xiaoqi Ren, Eric Malmi, Felix Weissenberger, Blanca Huergo, Maria Bauza, Thomas Lampe, Arthur Douillard, Mojtaba Seyedhosseini, Roy Frostig, Zoubin Ghahramani, Kelvin Nguyen, Kashyap Krishnakumar, Chengxi Ye, Rahul Gupta, Alireza Nazari, Robert Geirhos, Pete Shaw, Ahmed Eleryan, Dima Damen, Jennimaria Palomaki, Ted Xiao, Qiyin Wu, Quan Yuan, Phoenix Meadowlark, Matthew Bilotti, Raymond Lin, Mukund Sridhar, Yannick Schroecker, Da-Woon Chung, Jincheng Luo, Trevor Strohman, Tianlin Liu, Anne Zheng, Jesse Emond, Wei Wang, Andrew Lampinen, Toshiyuki Fukuzawa, Folawiyo Campbell-Ajala, Monica Roy, James Lee-Thorp, Lily Wang, Iftekhar Naim, Tony, Nguy ên, Guy Bensky, Aditya Gupta, Dominika Rogozińska, Justin Fu, Thanumalayan Sankaranarayana Pillai, Petar Veličković, Shahar Drath, Philipp Neubeck, Vaibhav Tulsyan, Arseniy Klimovskiy, Don Metzler, Sage Stevens, Angel Yeh, Junwei Yuan, Tianhe Yu, Kelvin Zhang, Alec Go, Vincent Tsang, Ying Xu, Andy Wan, Isaac Galatzer-Levy, Sam Sobell, Abodunrinwa Toki, Elizabeth Salesky, Wenlei Zhou, Diego Antognini, Sholto Douglas, Shimu Wu, Adam Lelkes, Frank Kim, Paul Cavallaro, Ana Salazar, Yuchi Liu, James Besley, Tiziana Refice, Yiling Jia, Zhang Li, Michal Sokolik, Arvind Kannan, Jon Simon, Jo Chick, Avia Aharon, Meet Gandhi, Mayank Daswani, Keyvan Amiri, Vighnesh Birodkar, Abe Ittycheriah, Peter Grabowski, Oscar Chang, Charles Sutton, Zhixin, Lai, Umesh Telang, Susie Sargsyan, Tao Jiang, Raphael Hoffmann, Nicole Brichtova, Matteo Hessel, Jonathan Halcrow, Sammy Jerome, Geoff Brown, Alex Tomala, Elena Buchatskaya, Dian Yu, Sachit Menon, Pol Moreno, Yuguo Liao, Vicky Zayats, Luming Tang, SQ Mah, Ashish Shenoy, Alex Siegman, Majid Hadian, Okwan Kwon, Tao Tu, Nima Khajehnouri, Ryan Foley, Parisa Haghani, Zhongru Wu, Vaishakh Keshava, Khyatti Gupta, Tony Bruguier, Rui Yao, Danny Karmon, Luisa Zintgraf, Zhicheng Wang, Enrique Piqueras, Junehyuk Jung, Jenny Brennan, Diego Machado, Marissa Giustina, MH Tessler, Kamyu Lee, Qiao Zhang, Joss Moore, Kaspar Daugaard, Alexander Frömmgen, Jennifer Beattie, Fred Zhang, Daniel Kasenberg, Ty Geri, Danfeng Qin, Gaurav Singh Tomar, Tom Ouyang, Tianli Yu, Luowei Zhou, Rajiv Mathews, Andy Davis, Yaoyiran Li, Jai Gupta, Damion Yates, Linda Deng, Elizabeth Kemp, Ga-Young Joung, Sergei Vassilvitskii, Mandy Guo, Pallavi LV, Dave Dopson, Sami Lachgar, Lara McConnaughey, Himadri Choudhury, Dragos Dena, Aaron Cohen, Joshua Ainslie, Sergey Levi, Parthasarathy Gopavarapu, Polina Zablotskaia, Hugo Vallet, Sanaz Bahargam, Xiaodan Tang, Nenad Tomasev, Ethan Dyer, Daniel Balle, Hongrae Lee, William Bono, Jorge Gonzalez Mendez, Vadim Zubov, Shentao Yang, Ivor Rendulic, Yanyan Zheng, Andrew Hogue, Golan Pundak, Ralph Leith, Avishkar Bhoopchand, Michael Han, Mislav Žanić, Tom Schaul, Manolis Delakis, Tejas Iyer, Guanyu Wang, Harman Singh, Abdelrahman Abdelhamed, Tara Thomas, Siddhartha Brahma, Hilal Dib, Naveen Kumar, Wenxuan Zhou, Liang Bai, Pushkar Mishra, Jiao Sun, Valentin Anklin, Roykrong Sukkerd, Lauren Agubuzu, Anton Briukhov, Anmol Gulati, Maximilian Sieb, Fabio Pardo, Sara Nasso, Junquan Chen, Kexin Zhu, Tiberiu Sosea, Alex Goldin, Keith Rush, Spurthi Amba Hombaiah, Andreas Noever, Allan Zhou, Sam Haves, Mary Phuong, Jake Ades, Yi ting Chen, Lin Yang, Joseph Pagadora, Stan Bileschi, Victor Cotruta, Rachel Saputro, Arijit Pramanik, Sean Ammirati, Dan Garrette, Kevin Villela, Tim Blyth, Canfer Akbulut, Neha Jha, Alban Rrustemi, Arissa Wongpanich, Chirag Nagpal, Yonghui Wu, Morgane Rivière, Sergey Kishchenko, Pranesh Srinivasan, Alice Chen, Animesh Sinha, Trang Pham, Bill Jia, Tom Hennigan, Anton Bakalov, Nithya Attaluri, Drew Garmon, Daniel Rodriguez, Dawid Wegner, Wenhao Jia, Evan Senter, Noah Fiedel, Denis Petek, Yuchuan Liu, Cassidy Hardin, Harshal Tushar Lehri, Joao Carreira, Sara Smoot, Marcel Prasetya, Nami Akazawa, Anca Stefanoiu, Chia-Hua Ho, Anelia Angelova, Kate Lin, Min Kim, Charles Chen, Marcin Sieniek, Alice Li, Tongfei Guo, Sorin Baltateanu, Pouya Tafti, Michael Wunder, Nadav Olmert, Divyansh Shukla, Jingwei Shen, Neel Kovelamudi, Balaji Venkatraman, Seth Neel, Romal Thoppilan, Jerome Connor, Frederik Benzing, Axel Stjerngren, Golnaz Ghiasi, Alex Polozov, Joshua Howland, Theophane Weber, Justin Chiu, Ganesh Poomal Girirajan, Andreas Terzis, Pidong Wang, Fangda Li, Yoav Ben Shalom, Dinesh Tewari, Matthew Denton, Roee Aharoni, Norbert Kalb, Heri Zhao, Junlin Zhang, Angelos Filos, Matthew Rahtz, Lalit Jain, Connie Fan, Vitor Rodrigues, Ruth Wang, Richard Shin, Jacob Austin, Roman Ring, Mariella Sanchez-Vargas, Mehadi Hassen, Ido Kessler, Uri Alon, Gufeng Zhang, Wenhu Chen, Yenai Ma, Xiance Si, Le Hou, Azalia Mirhoseini, Marc Wilson, Geoff Bacon, Becca Roelofs, Lei Shu, Gautam Vasudevan, Jonas Adler, Artur Dwornik, Tayfun Terzi, Matt Lawlor, Harry Askham, Mike Bernico, Xuanyi Dong, Chris Hidey, Kevin Kilgour, Gaël Liu, Surya Bhupatiraju, Luke Leonhard, Siqi Zuo, Partha Talukdar, Qing Wei, Aliaksei Severyn, Vít Listík, Jong Lee, Aditya Tripathi, SK Park, Yossi Matias, Hao Liu, Alex Ruiz, Rajesh Jayaram, Jackson Tolins, Pierre Marcenac, Yiming Wang, Bryan Seybold, Henry Prior, Deepak Sharma, Jack Weber, Mikhail Sirotenko, Yunhsuan Sung, Dayou Du, Ellie Pavlick, Stefan Zinke, Markus Freitag, Max Dylla, Montse Gonzalez Arenas, Natan Potikha, Omer Goldman, Connie Tao, Rachita Chhaparia, Maria Voitovich, Pawan Dogra, Andrija Ražnatović, Zak Tsai, Chong You, Oleaser Johnson, George Tucker, Chenjie Gu, Jae Yoo, Maryam Majzoubi, Valentin Gabeur, Bahram Raad, Rocky Rhodes, Kashyap Kolipaka, Heidi Howard, Geta Sampemane, Benny Li, Chulayuth Asawaroengchai, Duy Nguyen, Chiyuan Zhang, Timothee Cour, Xinxin Yu, Zhao Fu, Joe Jiang, Po-Sen Huang, Gabriela Surita, Iñaki Iturrate, Yael Karov, Michael Collins, Martin Baeuml, Fabian Fuchs, Shilpa Shetty, Swaroop Ramaswamy, Sayna Ebrahimi, Qiuchen Guo, Jeremy Shar, Gabe Barth-Maron, Sravanti Addepalli, Bryan Richter, Chin-Yi Cheng, Eugénie Rives, Fei Zheng, Johannes Griesser, Nishanth Dikkala, Yoel Zeldes, Ilkin Safarli, Dipanjan Das, Himanshu Srivastava, Sadh MNM Khan, Xin Li, Aditya Pandey, Larisa Markeeva, Dan Belov, Qiqi Yan, Mikołaj Rybiński, Tao Chen, Megha Nawhal, Michael Quinn, Vineetha Govindaraj, Sarah York, Reed Roberts, Roopal Garg, Namrata Godbole, Jake Abernethy, Anil Das, Lam Nguyen Thiet, Jonathan Tompson, John Nham, Neera Vats, Ben Caine, Wesley Helmholz, Francesco Pongetti, Yeongil Ko, James An, Clara Huiyi Hu, Yu-Cheng Ling, Julia Pawar, Robert Leland, Keisuke Kinoshita, Waleed Khawaja, Marco Selvi, Eugene Ie, Danila Sinopalnikov, Lev Proleev, Nilesh Tripuraneni, Michele Bevilacqua, Seungji Lee, Clayton Sanford, Dan Suh, Dustin Tran, Jeff Dean, Simon Baumgartner, Jens Heitkaemper, Sagar Gubbi, Kristina Toutanova, Yichong Xu, Chandu Thekkath, Keran Rong, Palak Jain, Annie Xie, Yan Virin, Yang Li, Lubo Litchev, Richard Powell, Tarun Bharti, Adam Kraft, Nan Hua, Marissa Ikonomidis, Ayal Hitron, Sanjiv Kumar, Loic Matthey, Sophie Bridgers, Lauren Lax, Ishaan Malhi, Ondrej Skopek, Ashish Gupta, Jiawei Cao, Mitchelle Rasquinha, Siim Põder, Wojciech Stokowiec, Nicholas Roth, Guowang Li, Michaël Sander, Joshua Kessinger, Vihan Jain, Edward Loper, Wonpyo Park, Michal Yarom, Liqun Cheng, Guru Guruganesh, Kanishka Rao, Yan Li, Catarina Barros, Mikhail Sushkov, Chun-Sung Ferng, Rohin Shah, Ophir Aharoni, Ravin Kumar, Tim McConnell, Peiran Li, Chen Wang, Fernando Pereira, Craig Swanson, Fayaz Jamil, Yan Xiong, Anitha Vijayakumar, Prakash Shroff, Kedar Soparkar, Jindong Gu, Livio Baldini Soares, Eric Wang, Kushal Majmundar, Aurora Wei, Kai Bailey, Nora Kassner, Chizu Kawamoto, Goran Žužić, Victor Gomes, Abhirut Gupta, Michael Guzman, Ishita Dasgupta, Xinyi Bai, Zhufeng Pan, Francesco Piccinno, Hadas Natalie Vogel, Octavio Ponce, Adrian Hutter, Paul Chang, Pan-Pan Jiang, Ionel Gog, Vlad Ionescu, James Manyika, Fabian Pedregosa, Harry Ragan, Zach Behrman, Ryan Mullins, Coline Devin, Aroonalok Pyne, Swapnil Gawde, Martin Chadwick, Yiming Gu, Sasan Tavakkol, Andy Twigg, Naman Goyal, Ndidi Elue, Anna Goldie, Srinivasan Venkatachary, Hongliang Fei, Ziqiang Feng, Marvin Ritter, Isabel Leal, Sudeep Dasari, Pei Sun, Alif Raditya Rochman, Brendan O'Donoghue, Yuchen Liu, Jim Sproch, Kai Chen, Natalie Clay, Slav Petrov, Sailesh Sidhwani, Ioana Mihailescu, Alex Panagopoulos, AJ Piergiovanni, Yunfei Bai, George Powell, Deep Karkhanis, Trevor Yacovone, Petr Mitrichev, Joe Kovac, Dave Uthus, Amir Yazdanbakhsh, David Amos, Steven Zheng, Bing Zhang, Jin Miao, Bhuvana Ramabhadran, Soroush Radpour, Shantanu Thakoor, Josh Newlan, Oran Lang, Orion Jankowski, Shikhar Bharadwaj, Jean-Michel Sarr, Shereen Ashraf, Sneha Mondal, Jun Yan, Ankit Singh Rawat, Sarmishta Velury, Greg Kochanski, Tom Eccles, Franz Och, Abhanshu Sharma, Ethan Mahintorabi, Alex Gurney, Carrie Muir, Vered Cohen, Saksham Thakur, Adam Bloniarz, Asier Mujika, Alexander Pritzel, Paul Caron, Altaf Rahman, Fiona Lang, Yasumasa Onoe, Petar Sirkovic, Jay Hoover, Ying Jian, Pablo Duque, Arun Narayanan, David Soergel, Alex Haig, Loren Maggiore, Shyamal Buch, Josef Dean, Ilya Figotin, Igor Karpov, Shaleen Gupta, Denny Zhou, Muhuan Huang, Ashwin Vaswani, Christopher Semturs, Kaushik Shivakumar, Yu Watanabe, Vinodh Kumar Rajendran, Eva Lu, Yanhan Hou, Wenting Ye, Shikhar Vashishth, Nana Nti, Vytenis Sakenas, Darren Ni, Doug DeCarlo, Michael Bendersky, Sumit Bagri, Nacho Cano, Elijah Peake, Simon Tokumine, Varun Godbole, Carlos Guía, Tanya Lando, Vittorio Selo, Seher Ellis, Danny Tarlow, Daniel Gillick, Alessandro Epasto, Siddhartha Reddy Jonnalagadda, Meng Wei, Meiyan Xie, Ankur Taly, Michela Paganini, Mukund Sundararajan, Daniel Toyama, Ting Yu, Dessie Petrova, Aneesh Pappu, Rohan Agrawal, Senaka Buthpitiya, Justin Frye, Thomas Buschmann, Remi Crocker, Marco Tagliasacchi, Mengchao Wang, Da Huang, Sagi Perel, Brian Wieder, Hideto Kazawa, Weiyue Wang, Jeremy Cole, Himanshu Gupta, Ben Golan, Seojin Bang, Nitish Kulkarni, Ken Franko, Casper Liu, Doug Reid, Sid Dalmia, Jay Whang, Kevin Cen, Prasha Sundaram, Johan Ferret, Berivan Isik, Lucian Ionita, Guan Sun, Anna Shekhawat, Muqthar Mohammad, Philip Pham, Ronny Huang, Karthik Raman, Xingyi Zhou, Ross Mcilroy, Austin Myers, Sheng Peng, Jacob Scott, Paul Covington, Sofia Erell, Pratik Joshi, João Gabriel Oliveira, Natasha Noy, Tajwar Nasir, Jake Walker, Vera Axelrod, Tim Dozat, Pu Han, Chun-Te Chu, Eugene Weinstein, Anand Shukla, Shreyas Chandrakaladharan, Petra Poklukar, Bonnie Li, Ye Jin, Prem Eruvbetine, Steven Hansen, Avigail Dabush, Alon Jacovi, Samrat Phatale, Chen Zhu, Steven Baker, Mo Shomrat, Yang Xiao, Jean Pouget-Abadie, Mingyang Zhang, Fanny Wei, Yang Song, Helen King, Yiling Huang, Yun Zhu, Ruoxi Sun, Juliana Vicente Franco, Chu-Cheng Lin, Sho Arora, Hui, Li, Vivian Xia, Luke Vilnis, Mariano Schain, Kaiz Alarakyia, Laurel Prince, Aaron Phillips, Caleb Habtegebriel, Luyao Xu, Huan Gui, Santiago Ontanon, Lora Aroyo, Karan Gill, Peggy Lu, Yash Katariya, Dhruv Madeka, Shankar Krishnan, Shubha Srinivas Raghvendra, James Freedman, Yi Tay, Gaurav Menghani, Peter Choy, Nishita Shetty, Dan Abolafia, Doron Kukliansky, Edward Chou, Jared Lichtarge, Ken Burke, Ben Coleman, Dee Guo, Larry Jin, Indro Bhattacharya, Victoria Langston, Yiming Li, Suyog Kotecha, Alex Yakubovich, Xinyun Chen, Petre Petrov, Tolly Powell, Yanzhang He, Corbin Quick, Kanav Garg, Dawsen Hwang, Yang Lu, Srinadh Bhojanapalli, Kristian Kjems, Ramin Mehran, Aaron Archer, Hado van Hasselt, Ashwin Balakrishna, JK Kearns, Meiqi Guo, Jason Riesa, Mikita Sazanovich, Xu Gao, Chris Sauer, Chengrun Yang, XiangHai Sheng, Thomas Jimma, Wouter Van Gansbeke, Vitaly Nikolaev, Wei Wei, Katie Millican, Ruizhe Zhao, Justin Snyder, Levent Bolelli, Maura O'Brien, Shawn Xu, Fei Xia, Wentao Yuan, Arvind Neelakantan, David Barker, Sachin Yadav, Hannah Kirkwood, Farooq Ahmad, Joel Wee, Jordan Grimstad, Boyu Wang, Matthew Wiethoff, Shane Settle, Miaosen Wang, Charles Blundell, Jingjing Chen, Chris Duvarney, Grace Hu, Olaf Ronneberger, Alex Lee, Yuanzhen Li, Abhishek Chakladar, Alena Butryna, Georgios Evangelopoulos, Guillaume Desjardins, Jonni Kanerva, Henry Wang, Averi Nowak, Nick Li, Alyssa Loo, Art Khurshudov, Laurent El Shafey, Nagabhushan Baddi, Karel Lenc, Yasaman Razeghi, Tom Lieber, Amer Sinha, Xiao Ma, Yao Su, James Huang, Asahi Ushio, Hanna Klimczak-Plucińska, Kareem Mohamed, JD Chen, Simon Osindero, Stav Ginzburg, Lampros Lamprou, Vasilisa Bashlovkina, Duc-Hieu

Tran, Ali Khodaei, Ankit Anand, Yixian Di, Ramy Eskander, Manish Reddy Vuyyuru, Jasmine Liu, Aishwarya Kamath, Roman Goldenberg, Mathias Bellaiche, Juliette Pluto, Bill Rosgen, Hassan Mansoor, William Wong, Suhas Ganesh, Eric Bailey, Scott Baird, Dan Deutsch, Jinoo Baek, Xuhui Jia, Chansoo Lee, Abe Friesen, Nathaniel Braun, Kate Lee, Amayika Panda, Steven M. Hernandez, Duncan Williams, Jianqiao Liu, Ethan Liang, Arnaud Autef, Emily Pitler, Deepali Jain, Phoebe Kirk, Oskar Bunyan, Jaume Sanchez Elias, Tongxin Yin, Machel Reid, Aedan Pope, Nikita Putikhin, Bidisha Samanta, Sergio Guadarrama, Dahun Kim, Simon Rowe, Marcella Valentine, Geng Yan, Alex Salcianu, David Silver, Gan Song, Richa Singh, Shuai Ye, Hannah DeBalsi, Majd Al Merey, Eran Ofek, Albert Webson, Shibl Mourad, Ashwin Kakarla, Silvio Lattanzi, Nick Roy, Evgeny Sluzhaev, Christina Butterfield, Alessio Tonioni, Nathan Waters, Sudhindra Kopalle, Jason Chase, James Cohan, Girish Ramchandra Rao, Robert Berry, Michael Voznesensky, Shuguang Hu, Kristen Chiafullo, Sharat Chikkerur, George Scrivener, Ivy Zheng, Jeremy Wiesner, Wolfgang Macherey, Timothy Lillicrap, Fei Liu, Brian Walker, David Welling, Elinor Davies, Yangsibo Huang, Lijie Ren, Nir Shabat, Alessandro Agostini, Mariko Iinuma, Dustin Zelle, Rohit Sathyanarayana, Andrea D'olimpio, Morgan Redshaw, Matt Ginsberg, Ashwin Murthy, Mark Geller, Tatiana Matejovicova, Ayan Chakrabarti, Ryan Julian, Christine Chan, Qiong Hu, Daniel Jarrett, Manu Agarwal, Jeshwanth Challagundla, Tao Li, Sandeep Tata, Wen Ding, Maya Meng, Zhuyun Dai, Giulia Vezzani, Shefali Garg, Jannis Bulian, Mary Jasarevic, Honglong Cai, Harish Rajamani, Adam Santoro, Florian Hartmann, Chen Liang, Bartek Perz, Apoorv Jindal, Fan Bu, Sungyong Seo, Ryan Poplin, Adrian Goedeckemeyer, Badih Ghazi, Nikhil Khadke, Leon Liu, Kevin Mather, Mingda Zhang, Ali Shah, Alex Chen, Jinliang Wei, Keshav Shivam, Yuan Cao, Donghyun Cho, Angelo Scorza Scarpati, Michael Moffitt, Clara Barbu, Ivan Jurin, Ming-Wei Chang, Hongbin Liu, Hao Zheng, Shachi Dave, Christine Kaeser-Chen, Xiaobin Yu, Alvin Abdagic, Lucas Gonzalez, Yanping Huang, Peilin Zhong, Cordelia Schmid, Bryce Petrini, Alex Wertheim, Jifan Zhu, Hoang Nguyen, Kaiyang Ji, Yanqi Zhou, Tao Zhou, Fangxiaoyu Feng, Regev Cohen, David Rim, Shubham Milind Phal, Petko Georgiev, Ariel Brand, Yue Ma, Wei Li, Somit Gupta, Chao Wang, Pavel Dubov, Jean Tarbouriech, Kingshuk Majumder, Huijian Li, Norman Rink, Apurv Suman, Yang Guo, Yinghao Sun, Arun Nair, Xiaowei Xu, Mohamed Elhawaty, Rodrigo Cabrera, Guangxing Han, Julian Eisenschlos, Junwen Bai, Yuqi Li, Yamini Bansal, Thibault Sellam, Mina Khan, Hung Nguyen, Justin Mao-Jones, Nikos Parotsidis, Jake Marcus, Cindy Fan, Roland Zimmermann, Yony Kochinski, Laura Graesser, Feryal Behbahani, Alvaro Caceres, Michael Riley, Patrick Kane, Sandra Lefdal, Rob Willoughby, Paul Vicol, Lun Wang, Shujian Zhang, Ashleah Gill, Yu Liang, Gautam Prasad, Soroosh Mariooryad, Mehran Kazemi, Zifeng Wang, Kritika Muralidharan, Paul Voigtlaender, Jeffrey Zhao, Huanjie Zhou, Nina D'Souza, Aditi Mavalankar, Séb Arnold, Nick Young, Obaid Sarvana, Chace Lee, Milad Nasr, Tingting Zou, Seokhwan Kim, Lukas Haas, Kaushal Patel, Neslihan Bulut, David Parkinson, Courtney Biles, Dmitry Kalashnikov, Chi Ming To, Aviral Kumar, Jessica Austin, Alex Greve, Lei Zhang, Megha Goel, Yeqing Li, Sergey Yaroshenko, Max Chang, Abhishek Jindal, Geoff Clark, Hagai Taitelbaum, Dale Johnson, Ofir Roval, Jeongwoo Ko, Anhad Mohananey, Christian Schuler, Shenil Dodhia, Ruichao Li, Kazuki Osawa, Claire Cui, Peng Xu, Rushin Shah, Tao Huang, Ela Gruzewska, Nathan Clement, Mudit Verma, Olcan Sercinoglu, Hai Qian, Viral Shah, Masa Yamaguchi, Abhinit Modi, Takahiro Kosakai, Thomas Strohmann, Junhao Zeng, Beliz Gunel, Jun Qian, Austin Tarango, Krzysztof Jastrzębski, Robert David, Jyn Shan, Parker Schuh, Kunal Lad, Willi Gierke, Mukundan Madhavan, Xinyi Chen, Mark Kurzeja, Rebeca Santamaria-Fernandez, Dawn Chen, Alexandra Cordell, Yuri Chervonyi, Frankie Garcia, Nithish Kannen, Vincent Perot, Nan Ding, Shlomi Cohen-Ganor, Victor Lavrenko, Junru Wu, Georgie Evans, Cicero Nogueira dos Santos, Madhavi Sewak, Ashley Brown, Andrew Hard, Joan Puigcerver, Zeyu Zheng, Yizhong Liang, Evgeny Gladchenko, Reeve Ingle, Uri First, Pierre Sermanet, Charlotte Magister, Mihajlo Velimirović, Sashank Reddi, Susanna Ricco, Eirikur Agustsson, Hartwig Adam, Nir Levine, David Gaddy, Dan Holtmann-Rice, Xuanhui Wang, Ashutosh Sathe, Abhijit Guha Roy, Blaž Bratanič, Alen Carin, Harsh Mehta, Silvano Bonacina, Nicola De Cao, Mara Finkelstein, Verena Rieser, Xinyi Wu, Florent Altché, Dylan Scandinaro, Li Li, Nino Vieillard, Nikhil Sethi, Garrett Tanzer, Zhi Xing, Shibo Wang, Parul Bhatia, Gui Citovsky, Thomas Anthony, Sharon Lin, Tianze Shi, Shoshana Jakobovits, Gena Gibson, Raj Apte, Lisa Lee, Mingqing Chen, Arunkumar Byravan, Petros Maniatis, Kellie Webster, Andrew Dai, Pu-Chin Chen, Jiaqi Pan, Asya Fadeeva, Zach Gleicher, Thang Luong, and Niket Kumar Bhumihar. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities, 2025. URL `https://arxiv.org/abs/2507.06261`.

Emma Croxford, Yanjun Gao, Elliot First, Nicholas Pellegrino, Miranda Schnier, John Caskey, Max Oguss, Graham Wills, Guanhua Chen, Dmitriy Dligach, Matthew M. Churpek, Anoop Mayampurath, Frank Liao, Cherodeep Goswami, Karen K. Wong, Brian W. Patterson, and Majid Afshar. Automating evaluation of ai text generation in healthcare with a large language model (llm)-as-a-judge. *medRxiv*, May 2025. doi: 10.1101/2025.04.22.25326219. URL https://www.medrxiv.org/content/10.1101/2025.04.22.25326219v1. Preprint.

Leonardo Mendonça de Moura and Sebastian Ullrich. The lean 4 theorem prover and programming language. In *CADE*, 2021. URL https://api.semanticscholar.org/CorpusID:235800962.

Leonardo Mendonça de Moura, Soonho Kong, Jeremy Avigad, Floris van Doorn, and Jakob von Raumer. The lean theorem prover (system description). In *CADE*, 2015. URL https://api.semanticscholar.org/CorpusID:232990.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

Jasper Dekoninck, Ivo Petrov, Kristian Minchev, Mislav Balunovic, Martin Vechev, Miroslav Marinov, Maria Drencheva, Lyuba Konova, Milen Shumanov, Kaloyan Tsvetkov, Nikolay Drenchev, Lazar Todorov, Kalina Nikolova, Nikolay Georgiev, Vanesa Kalinkova, and Margulan Ismoldayev. The open proof corpus: A large-scale study of llm-generated mathematical proofs, 2025. URL https://arxiv.org/abs/2506.21621.

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

Dadi Guo, Jiayu Liu, Zhiyuan Fan, Zhitao He, Haoran Li, Yumeng Wang, and Yi R. Fung. Mathematical proof as a litmus test: Revealing failure modes of advanced large reasoning models, 2025. URL https://arxiv.org/abs/2506.17114.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning, 2020. URL https://arxiv.org/abs/1911.05722.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset, 2021. URL https://arxiv.org/abs/2103.03874.

HMMT. The harvard-mit mathematics tournament (hmmt). https://www.hmmt.org/, 2025. Accessed: 2025-09-14.

Jeremy Howard. The /llms.txt file: A proposal to help llms use websites, September 2024. URL https://llmstxt.org/. A proposal to standardize the use of /llms.txt files for providing LLM-friendly content on websites.

Gérard P. Huet and Christine Paulin-Mohring. The coq proof assistant reference manual. 2000. URL https://api.semanticscholar.org/CorpusID:59695435.

Maxwell Jia. Aime 2024 dataset. https://huggingface.co/datasets/Maxwell-Jia/AIME_2024, feb 2025. American Invitational Mathematics Examination 2024, 30 items.

Seungone Kim, Jamin Shin, Yejin Cho, Joel Jang, Shayne Longpre, Hwaran Lee, Sangdoo Yun, Seongjin Shin, Sungdong Kim, James Thorne, and Minjoon Seo. Prometheus: Inducing fine-grained evaluation capability in language models, 2024a. URL https://arxiv.org/abs/2310.08491.

Seungone Kim, Juyoung Suk, Shayne Longpre, Bill Yuchen Lin, Jamin Shin, Sean Welleck, Graham Neubig, Moontae Lee, Kyungjae Lee, and Minjoon Seo. Prometheus 2: An open source language model specialized in evaluating other language models, 2024b. URL https://arxiv.org/abs/2405.01535.

Aitor Lewkowycz, Anders Johan Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Venkatesh Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. Solving quantitative reasoning problems with language models. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=IFXTZERXdM7.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=v8L0pN6EOi.

Yong Lin, Shange Tang, Bohan Lyu, Jiayun Wu, Hongzhou Lin, Kaiyu Yang, Jia Li, Mengzhou Xia, Danqi Chen, Sanjeev Arora, and Chi Jin. Goedel-prover: A frontier model for open-source automated theorem proving, 2025. URL https://arxiv.org/abs/2502.07640.

Chengwu Liu, Jianhao Shen, Huajian Xin, Zhengying Liu, Ye Yuan, Haiming Wang, Wei Ju, Chuanyang Zheng, Yichun Yin, Lin Li, Ming Zhang, and Qun Liu. Fimo: A challenge formal dataset for automated theorem proving, 2023. URL https://arxiv.org/abs/2309.04295.

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. Combibench: Benchmarking llm capability for combinatorial mathematics, 2025. URL https://arxiv.org/abs/2505.03171.

MAA. American invitational mathematics examination (aime). https://maa.org/math-competitions/aime, 2025. Accessed: 2025-09-14.

Mistral-AI, :, Abhinav Rastogi, Albert Q. Jiang, Andy Lo, Gabrielle Berrada, Guillaume Lample, Jason Rute, Joep Barmentlo, Karmesh Yadav, Kartik Khandelwal, Khyathi Raghavi Chandu, Léonard Blier, Lucile Saulnier, Matthieu Dinot, Maxime Darrin, Neha Gupta, Roman Soletskyi, Sagar Vaze, Teven Le Scao, Yihan Wang, Adam Yang, Alexander H. Liu, Alexandre Sablayrolles, Amélie Héliou, Amélie Martin, Andy Ehrenberg, Anmol Agarwal, Antoine Roux, Arthur Darcet, Arthur Mensch, Baptiste Bout, Baptiste Rozière, Baudouin De Monicault, Chris Bamford, Christian Wallenwein, Christophe Renaudin, Clémence Lanfranchi, Darius Dabert, Devon

Mizelle, Diego de las Casas, Elliot Chane-Sane, Emilien Fugier, Emma Bou Hanna, Gauthier Delerce, Gauthier Guinet, Georgii Novikov, Guillaume Martin, Himanshu Jaju, Jan Ludziejewski, Jean-Hadrien Chabran, Jean-Malo Delignon, Joachim Studnia, Jonas Amar, Josselin Somerville Roberts, Julien Denize, Karan Saxena, Kush Jain, Lingxiao Zhao, Louis Martin, Luyu Gao, Lélio Renard Lavaud, Marie Pellat, Mathilde Guillaumin, Mathis Felardos, Maximilian Augustin, Mickaël Seznec, Nikhil Raghuraman, Olivier Duchenne, Patricia Wang, Patrick von Platen, Patryk Saffer, Paul Jacob, Paul Wambergue, Paula Kurylowicz, Pavankumar Reddy Muddireddy, Philomène Chagniot, Pierre Stock, Pravesh Agrawal, Romain Sauvestre, Rémi Delacourt, Sanchit Gandhi, Sandeep Subramanian, Shashwat Dalal, Siddharth Gandhi, Soham Ghosh, Srijan Mishra, Sumukh Aithal, Szymon Antoniak, Thibault Schueller, Thibaut Lavril, Thomas Robert, Thomas Wang, Timothée Lacroix, Valeriia Nemychnikova, Victor Paltz, Virgile Richard, Wen-Ding Li, William Marshall, Xuanyu Zhang, and Yunhao Tang. Magistral, 2025. URL https://arxiv.org/abs/2506.10910.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling, 2025. URL https://arxiv.org/abs/2501.19393.

Tobias Nipkow, Markus Wenzel, and Lawrence C. Paulson. *Isabelle/HOL: A Proof Assistant for Higher-Order Logic*, volume 2283 of *Lecture Notes in Computer Science*. Springer, 2002. ISBN 978-3-540-43376-7. doi: 10.1007/3-540-45949-9.

OpenAI, :, Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, Alex Iftimie, Alex Karpenko, Alex Tachard Passos, Alexander Neitz, Alexander Prokofiev, Alexander Wei, Allison Tam, Ally Bennett, Ananya Kumar, Andre Saraiva, Andrea Vallone, Andrew Duberstein, Andrew Kondrich, Andrey Mishchenko, Andy Applebaum, Angela Jiang, Ashvin Nair, Barret Zoph, Behrooz Ghorbani, Ben Rossen, Benjamin Sokolowsky, Boaz Barak, Bob McGrew, Borys Minaiev, Botao Hao, Bowen Baker, Brandon Houghton, Brandon McKinzie, Brydon Eastman, Camillo Lugaresi, Cary Bassin, Cary Hudson, Chak Ming Li, Charles de Bourcy, Chelsea Voss, Chen Shen, Chong Zhang, Chris Koch, Chris Orsinger, Christopher Hesse, Claudia Fischer, Clive Chan, Dan Roberts, Daniel Kappler, Daniel Levy, Daniel Selsam, David Dohan, David Farhi, David Mely, David Robinson, Dimitris Tsipras, Doug Li, Dragos Oprica, Eben Freeman, Eddie Zhang, Edmund Wong, Elizabeth Proehl, Enoch Cheung, Eric Mitchell, Eric Wallace, Erik Ritter, Evan Mays, Fan Wang, Felipe Petroski Such, Filippo Raso, Florencia Leoni, Foivos Tsimpourlas, Francis Song, Fred von Lohmann, Freddie Sulit, Geoff Salmon, Giambattista Parascandolo, Gildas Chabot, Grace Zhao, Greg Brockman, Guillaume Leclerc, Hadi Salman, Haiming Bao, Hao Sheng, Hart Andrin, Hessam Bagherinezhad, Hongyu Ren, Hunter Lightman, Hyung Won Chung, Ian Kivlichan, Ian O'Connell, Ian Osband, Ignasi Clavera Gilaberte, Ilge Akkaya, Ilya Kostrikov, Ilya Sutskever, Irina Kofman, Jakub Pachocki, James Lennon, Jason Wei, Jean Harb, Jerry Twore, Jiacheng Feng, Jiahui Yu, Jiayi Weng, Jie Tang, Jieqi Yu, Joaquin Quiñonero Candela, Joe Palermo, Joel Parish, Johannes Heidecke, John Hallman, John Rizzo, Jonathan Gordon, Jonathan Uesato, Jonathan Ward, Joost Huizinga, Julie Wang, Kai Chen, Kai Xiao, Karan Singhal, Karina Nguyen, Karl Cobbe, Katy Shi, Kayla Wood, Kendra Rimbach, Keren Gu-Lemberg, Kevin Liu, Kevin Lu, Kevin Stone, Kevin Yu, Lama Ahmad, Lauren Yang, Leo Liu, Leon Maksin, Leyton Ho, Liam Fedus, Lilian Weng, Linden Li, Lindsay McCallum, Lindsey Held, Lorenz Kuhn, Lukas Kondraciuk, Lukasz Kaiser, Luke Metz, Madelaine Boyd, Maja Trebacz, Manas Joglekar, Mark Chen, Marko Tintor, Mason Meyer, Matt Jones, Matt Kaufer, Max Schwarzer, Meghan Shah, Mehmet Yatbaz, Melody Y. Guan, Mengyuan Xu, Mengyuan Yan, Mia Glaese, Mianna Chen, Michael Lampe, Michael Malek, Michele Wang, Michelle Fradin, Mike McClay, Mikhail Pavlov, Miles Wang, Mingxuan Wang, Mira Murati, Mo Bavarian, Mostafa Rohaninejad, Nat McAleese, Neil Chowdhury, Neil Chowdhury, Nick Ryder, Nikolas Tezak, Noam Brown, Ofir Nachum, Oleg Boiko, Oleg Murk, Olivia Watkins, Patrick Chao, Paul Ashbourne, Pavel Izmailov, Peter Zhokhov, Rachel Dias, Rahul Arora, Randall Lin, Rapha Gontijo Lopes, Raz Gaon, Reah Miyara, Reimar Leike, Renny Hwang, Rhythm Garg, Robin Brown, Roshan James, Rui Shu, Ryan Cheu, Ryan Greene, Saachi Jain, Sam Altman, Sam Toizer, Sam Toyer, Samuel Miserendino, Sandhini Agarwal, Santiago Hernandez, Sasha Baker, Scott McKinney, Scottie Yan, Shengjia Zhao, Shengli Hu, Shibani Santurkar, Shraman Ray Chaudhuri, Shuyuan Zhang, Siyuan Fu, Spencer Papay, Steph Lin, Suchir Balaji, Suvansh Sanjeev, Szymon Sidor, Tal Broda, Aidan Clark, Tao Wang, Taylor Gordon, Ted Sanders, Tejal Patwardhan, Thibault Sottiaux, Thomas Degry, Thomas Dimson, Tianhao Zheng, Timur Garipov, Tom Stasi, Trapit Bansal, Trevor Creech, Troy Peterson, Tyna Eloundou, Valerie Qi, Vineet Kosaraju, Vinnie Monaco, Vitchyr Pong, Vlad Fomenko, Weiyi Zheng, Wenda Zhou, Wes McCabe, Wojciech Zaremba, Yann Dubois, Yinghai Lu, Yining Chen, Young Cha, Yu Bai, Yuchen He, Yuchen Zhang, Yunyun Wang, Zheng Shao, and Zhuohan Li. Openai o1 system card, 2024. URL https://arxiv.org/abs/2412.16720.

OpenCompass Team. Aime 2025 dataset. https://huggingface.co/datasets/opencompass/AIME2025, feb 2025. American Invitational Mathematics Examination 2025, 30 items.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback, 2022.

Stanislas Polu and Ilya Sutskever. Generative language modeling for automated theorem proving, 2020. URL https://arxiv.org/abs/2009.03393.

Stanislas Polu, Jesse Michael Han, Kunhao Zheng, Mantas Baksys, Igor Babuschkin, and Ilya Sutskever. Formal mathematics statement curriculum learning, 2022. URL https://arxiv.org/abs/2202.01344.

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

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. URL https://arxiv.org/abs/2504.21801.

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.

Jiayi Sheng, Luna Lyu, Jikai Jin, Tony Xia, Alex Gu, James Zou, and Pan Lu. Solving inequality proofs with large language models, 2025. URL https://arxiv.org/abs/2506.07927.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, Gaël Liu, Francesco Visin, Kathleen Kenealy, Lucas Beyer, Xiaohai Zhai, Anton Tsitsulin, Robert Busa-Fekete, Alex Feng, Noveen Sachdeva, Benjamin Coleman, Yi Gao, Basil Mustafa, Iain Barr, Emilio Parisotto, David Tian, Matan Eyal, Colin Cherry, Jan-Thorsten Peter, Danila Sinopalnikov, Surya Bhupatiraju, Rishabh Agarwal, Mehran Kazemi, Dan Malkin, Ravin Kumar, David Vilar, Idan Brusilovsky, Jiaming Luo, Andreas Steiner, Abe Friesen, Abhanshu Sharma, Abheesht Sharma, Adi Mayrav Gilady, Adrian Goedeckemeyer, Alaa Saade, Alex Feng, Alexander Kolesnikov, Alexei Bendebury, Alvin Abdagic, Amit Vadi, András György, André Susano Pinto, Anil Das, Ankur Bapna, Antoine Miech, Antoine Yang, Antonia Paterson, Ashish Shenoy, Ayan Chakrabarti, Bilal Piot, Bo Wu, Bobak Shahriari, Bryce Petrini, Charlie Chen, Charline Le Lan, Christopher A. Choquette-Choo, CJ Carey, Cormac Brick, Daniel Deutsch, Danielle Eisenbud, Dee Cattle, Derek Cheng, Dimitris Paparas, Divyashree Shivakumar Sreepathihalli, Doug Reid, Dustin Tran, Dustin Zelle, Eric Noland, Erwin Huizenga, Eugene Kharitonov, Frederick Liu, Gagik Amirkhanyan, Glenn Cameron, Hadi Hashemi, Hanna Klimczak-Plucińska, Harman Singh, Harsh Mehta, Harshal Tushar Lehri, Hussein Hazimeh, Ian Ballantyne, Idan Szpektor, Ivan Nardini, Jean Pouget-Abadie, Jetha Chan, Joe Stanton, John Wieting, Jonathan Lai, Jordi Orbay, Joseph Fernandez, Josh Newlan, Ju yeong Ji, Jyotinder Singh, Kat Black, Kathy Yu, Kevin Hui, Kiran Vodrahalli, Klaus Greff, Linhai Qiu, Marcella Valentine, Marina Coelho, Marvin Ritter, Matt Hoffman, Matthew Watson, Mayank Chaturvedi, Michael Moynihan, Min Ma, Nabila Babar, Natasha Noy, Nathan Byrd, Nick Roy, Nikola Momchev, Nilay Chauhan, Noveen Sachdeva, Oskar Bunyan, Pankil Botarda, Paul Caron, Paul Kishan Rubenstein, Phil Culliton, Philipp Schmid, Pier Giuseppe Sessa, Pingmei Xu, Piotr Stanczyk, Pouya Tafti, Rakesh Shivanna, Renjie Wu, Renke Pan, Reza Rokni, Rob Willoughby, Rohith Vallu, Ryan Mullins, Sammy Jerome, Sara Smoot, Sertan Girgin, Shariq Iqbal, Shashir Reddy, Shruti Sheth, Siim Põder, Sijal Bhatnagar, Sindhu Raghuram Panyam, Sivan Eiger, Susan Zhang, Tianqi Liu, Trevor Yacovone, Tyler Liechty, Uday Kalra, Utku Evci, Vedant Misra, Vincent Roseberry, Vlad Feinberg, Vlad Kolesnikov, Woohyun Han, Woosuk Kwon, Xi Chen, Yinlam Chow, Yuvein Zhu, Zichuan Wei, Zoltan Egyed, Victor Cotruta, Minh Giang, Phoebe Kirk, Anand Rao, Kat Black, Nabila Babar, Jessica Lo, Erica Moreira, Luiz Gustavo Martins, Omar Sanseviero, Lucas Gonzalez, Zach Gleicher, Tris Warkentin, Vahab Mirrokni, Evan Senter, Eli Collins, Joelle Barral, Zoubin Ghahramani, Raia Hadsell, Yossi Matias, D. Sculley, Slav Petrov, Noah Fiedel, Noam Shazeer, Oriol Vinyals, Jeff Dean, Demis Hassabis, Koray Kavukcuoglu, Clement Farabet, Elena Buchatskaya, Jean-Baptiste Alayrac, Rohan Anil, Dmitry, Lepikhin, Sebastian Borgeaud, Olivier Bachem, Armand Joulin, Alek Andreev, Cassidy Hardin, Robert Dadashi, and Léonard Hussenot. Gemma 3 technical report, 2025. URL https://arxiv.org/abs/2503.19786.

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. doi: 10.1038/s41586-023-06747-5. URL https://doi.org/10.1038/s41586-023-06747-5.

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, 2024. URL https://arxiv.org/abs/2407.11214.

Haiming Wang, Mert Unsal, Xiaohan Lin, Mantas Baksys, Junqi Liu, Marco Dos Santos, Flood Sung, Marina Vinyes, Zhenzhe Ying, Zekai Zhu, Jianqiao Lu, Hugues de Saxcé, Bolton Bailey, Chendong Song, Chenjun Xiao, Dehao Zhang, Ebony Zhang, Frederick Pu, Han Zhu, Jiawei Liu, Jonas Bayer, Julien Michel, Longhui Yu, Léo Dreyfus-Schmidt, Lewis Tunstall, Luigi Pagani, Moreira Machado, Pauline Bourigault, Ran Wang, Stanislas Polu, Thibaut Barroyer, Wen-Ding Li, Yazhe Niu, Yann Fleureau, Yangyang Hu, Zhouliang Yu, Zihan Wang, Zhilin Yang, Zhengying Liu, and Jia Li. Kimina-prover preview: Towards large formal reasoning models with reinforcement learning, 2025a. URL https://arxiv.org/abs/2504.11354.

Peiyi Wang, Lei Li, Zhihong Shao, R. X. Xu, Damai Dai, Yifei Li, Deli Chen, Y. Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce llms step-by-step without human annotations, 2024a. URL https://arxiv.org/abs/2312.08935.

Yidong Wang, Zhuohao Yu, Zhengran Zeng, Linyi Yang, Cunxiang Wang, Hao Chen, Chaoya Jiang, Rui Xie, Jindong Wang, Xing Xie, Wei Ye, Shikun Zhang, and Yue Zhang. Pandalm: An automatic evaluation benchmark for llm instruction tuning optimization, 2024b. URL https://arxiv.org/abs/2306.05087.

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

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023. URL https://arxiv.org/abs/2201.11903.

Jingyuan Zhang, Qi Wang, Xingguang Ji, Yahui Liu, Yang Yue, Fuzheng Zhang, Di Zhang, Guorui Zhou, and Kun Gai. Leanabell-prover: Posttraining scaling in formal reasoning, 2025a. URL https://arxiv.org/abs/2504.06122.

Ziyin Zhang, Jiahao Xu, Zhiwei He, Tian Liang, Qiuzhi Liu, Yansi Li, Linfeng Song, Zhen-Pu Liang, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. Deeptheorem: Advancing llm reasoning for theorem proving through natural language and reinforcement learning. *ArXiv*, abs/2505.23754, 2025b. URL https://api.semanticscholar.org/CorpusID:278996747.

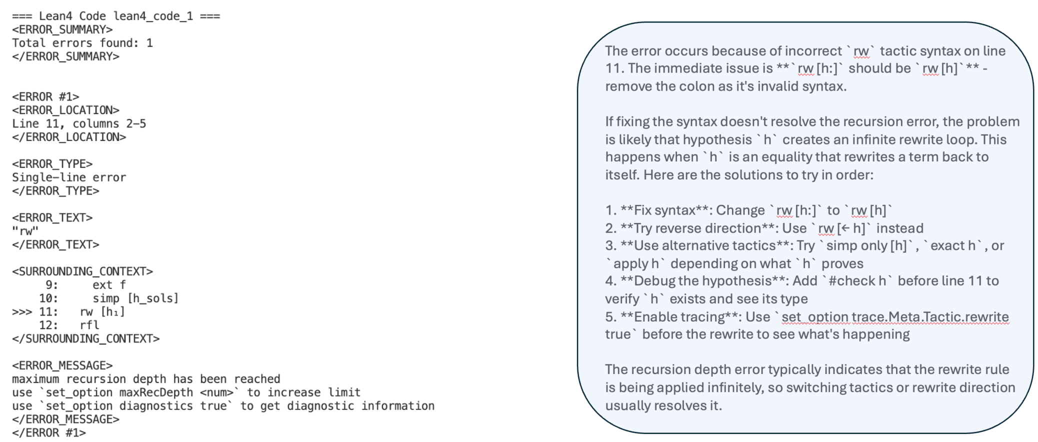
Yuwei Zhao, Ziyang Luo, Yuchen Tian, Hongzhan Lin, Weixiang Yan, Annan Li, and Jing Ma. Codejudge-eval: Can large language models be good judges in code understanding?, 2024. URL https://arxiv.org/abs/2408.10718.

Kunhao Zheng, Jesse Michael Han, and Stanislas Polu. Minif2f: a cross-system benchmark for formal olympiad-level mathematics, 2022. URL https://arxiv.org/abs/2109.00110.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena, 2023. URL https://arxiv.org/abs/2306.05685.

A USE OF LLMS

We used Grammarly's AI assistance to assist with grammar checking and sentence-level polishing.



(a) Lean4 Compilation Results. (b) Feedback generated by PROOF-VERIFIER.

Figure 6: Error analysis and feedback generation comparison.

B THEORETICAL ANALYSIS

B.1 CONSISTENCY-CONSTRAINED TRAINING

The consistency constraint in our Stage 1 reward function is motivated by a fundamental principle in mathematical evaluation: independent correct assessments of the same proof should converge to similar conclusions. To formalize this intuition, we begin by establishing the mathematical framework.

Given N parallel samples $\{r_1, r_2, \ldots, r_N\}$ for a proof attempt, each response r_i produces a binary prediction $p_i \in \{\text{True}, \text{False}\}$ and evaluation score $s_i \in [0, 100]$. Let $y \in \{\text{True}, \text{False}\}$ denote the ground truth label. We model the policy as $\pi_\theta(r_i|x)$ where x represents the input proof attempt.

Our fundamental assumption is that for a well-trained evaluator, the score distribution conditioned on correct predictions should concentrate around some true value. Formally, if we denote by $\mathcal{C} = \{r_i : p_i = y\}$ the set of correct predictions, then as the model quality improves, we expect:

$$s_i|p_i = y \xrightarrow{d} \delta_{\mu^*} \tag{4}$$

where δ_{μ^*} is a point mass at the true evaluation score μ^*, and $\xrightarrow{d}$ denotes convergence in distribution.

However, during training, we observe significant variance in scores even among correct predictions. To address this, we employ a consistency constraint based on modal consensus. Consider the empirical distribution of scores among correct predictions:

$$P_{\text{emp}}(s) = \frac{1}{|\mathcal{C}|} \sum_{r_i \in \mathcal{C}} \mathbf{1}[s_i = s] \tag{5}$$

For any estimator $\hat{s}$ of the consensus score, we can define the 0-1 consensus risk as:

$$R(\hat{s}) = \mathbb{E}[\mathbf{1}[s \neq \hat{s}]|p = y] = \sum_{s'} P(s = s'|p = y)\mathbf{1}[s' \neq \hat{s}] \tag{6}$$

Expanding this expression:

$$R(\hat{s}) = \sum_{s' \neq \hat{s}} P(s = s'|p = y) \tag{7}$$

$$= 1 - P(s = \hat{s}|p = y) \tag{8}$$

The mode estimator $\hat{s}_{\text{mode}} = \arg\max_s P(s|p = y)$ minimizes this risk since:

$$\hat{s}_{\text{mode}} = \arg\max_s P(s|p = y) = \arg\min_s [1 - P(s|p = y)] = \arg\min_s R(\hat{s}) \tag{9}$$

Furthermore, the mode exhibits superior robustness properties. Under ϵ-contamination where a fraction ϵ of the correct predictions are replaced by adversarial scores, the empirical distribution becomes:

$$P_{\text{cont}}(s) = (1 - \epsilon)P_{\text{emp}}(s) + \epsilon P_{\text{adv}}(s) \tag{10}$$

where $P_{\text{adv}}(s)$ is the adversarial distribution. The mode remains stable as long as $\epsilon < \frac{1}{2} - \frac{1}{2|\mathcal{S}|}$ where $|\mathcal{S}|$ is the number of distinct score values, while the sample mean can be arbitrarily shifted by any $\epsilon > 0$.

Now we analyze our reward function. The consistency-constrained reward can be written as:

$$R_1(r_i) = \mathbf{1}[p_i = y] \cdot (2\mathbf{1}[s_i = s_{\text{mode}}] - 1) - \mathbf{1}[p_i \neq y] \tag{11}$$

To understand the expected behavior, we compute the expected reward. Let $A = \{p_i = y\}$ denote the accuracy event and $C = \{s_i = s_{\text{mode}}\}$ denote the consistency event. Then:

$$\mathbb{E}[R_1(r_i)] = \mathbb{E}[\mathbf{1}[A] \cdot (2\mathbf{1}[C] - 1)] - \mathbb{E}[\mathbf{1}[A^c]] \tag{12}$$

$$= \mathbb{E}[\mathbf{1}[A] \cdot 2\mathbf{1}[C]] - \mathbb{E}[\mathbf{1}[A]] - P(A^c) \tag{13}$$

$$= 2\mathbb{E}[\mathbf{1}[A \cap C]] - P(A) - P(A^c) \tag{14}$$

$$= 2P(A \cap C) - P(A) - (1 - P(A)) \tag{15}$$

$$= 2P(A \cap C) - 1 \tag{16}$$

Using the conditional probability identity $P(A \cap C) = P(C|A)P(A)$:

$$\mathbb{E}[R_1(r_i)] = 2P(C|A)P(A) - 1 \tag{17}$$

$$= P(A)[2P(C|A) - \frac{1}{P(A)}] \tag{18}$$

$$= P(A)[2P(s_i = s_{\text{mode}}|p_i = y) - \frac{1}{P(A)}] \tag{19}$$

For the expected reward to be positive, we need:

$$P(s_i = s_{\text{mode}}|p_i = y) > \frac{1}{2P(A)} \tag{20}$$

This inequality reveals a crucial trade-off: when accuracy $P(A)$ is low, the consistency requirement becomes more stringent. However, as accuracy improves ($P(A) \to 1$), the consistency threshold approaches $\frac{1}{2}$, making positive rewards more achievable.

To analyze the training dynamics, consider the policy gradient:

$$\nabla_\theta \mathbb{E}[R_1(r_i)] = \mathbb{E}[R_1(r_i)\nabla_\theta \log \pi_\theta(r_i)] \tag{21}$$

Expanding using our reward decomposition:

$$\begin{aligned} \nabla_\theta \mathbb{E}[R_1(r_i)] = \mathbb{E}[\mathbf{1}[p_i = y](2\mathbf{1}[s_i = s_{\text{mode}}] - 1)\nabla_\theta \log \pi_\theta(r_i)] \quad &(22) \\ - \mathbb{E}[\mathbf{1}[p_i \neq y]\nabla_\theta \log \pi_\theta(r_i)] \quad &(23) \end{aligned}$$

This gradient has two components. The first term encourages both accuracy and consistency simultaneously, while the second term discourages incorrect predictions. The key insight is that the gradient magnitude for consistency is proportional to the accuracy level, creating a self-reinforcing dynamic.

As training progresses, we expect the accuracy $P(p_i = y)$ to increase. When $P(p_i = y) \to 1$, the expected reward simplifies to:

$$\lim_{P(p_i=y)\to 1} \mathbb{E}[R_1(r_i)] = 2P(s_i = s_{\text{mode}}|p_i = y) - 1 \tag{24}$$

At this stage, the training objective becomes purely a coordination problem: all correct predictions must agree on the modal score to achieve positive reward. The equilibrium of this coordination game occurs when all correct predictions produce the same score, i.e., when there exists a unique score s^* such that:

$$\pi_\theta(s = s^*|p = y) = 1 \quad \text{and} \quad \pi_\theta(s \neq s^*|p = y) = 0 \tag{25}$$

This implies that $\text{Var}(s_i|p_i = y) = 0$ at equilibrium, achieving perfect consistency among correct predictions.

Define the Lyapunov function $V(\theta) = -\mathbb{E}[R_1(r_i)]$. Along the policy gradient trajectory:

$$\begin{aligned} \frac{dV}{dt} &= -\nabla_\theta \mathbb{E}[R_1(r_i)]^T \frac{d\theta}{dt} \quad &(26) \\ &= -\alpha \|\nabla_\theta \mathbb{E}[R_1(r_i)]\|^2 \leq 0 \quad &(27) \end{aligned}$$

where $\alpha > 0$ is the learning rate. Since $R_1(r_i)$ is bounded, $V(\theta)$ is bounded below, ensuring convergence to a critical point with the desired properties.

B.2 Feedback Optimization via Score Improvement

The effectiveness of our Stage 2 training relies on a fundamental insight about the autoregressive generation process and how score improvements can serve as proxies for feedback quality. Our model generates responses following a specific sequential order: error analysis → evaluation score → feedback → final label.

Let $r = (a, s, f, p)$ denote a complete response where a represents error analysis, s the evaluation score, f the feedback, and p the final binary prediction. Under autoregressive factorization, the likelihood decomposes as:

$$\pi_\theta(r|x) = \pi_\theta(a|x)\pi_\theta(s|a, x)\pi_\theta(f|s, a, x)\pi_\theta(p|f, s, a, x) \tag{28}$$

The key observation is that since s is generated before f, the evaluation score represents the model's assessment of proof quality based purely on error analysis, independent of the feedback content. This temporal independence allows us to interpret score improvements as objective measures of feedback effectiveness.

Consider the following formalization: let τ_0 be an initial proof attempt and F_θ our feedback provider. When F_θ generates feedback f, a prover P uses this feedback to produce a revised proof τ_1. If our scoring function is S_ϕ, then the score improvement is $\Delta s = S_\phi(\tau_1) - S_\phi(\tau_0)$.

The fundamental assumption underlying our approach is that effective feedback should systematically lead to better proofs. Formally, for high-quality feedback f, we expect $\mathbb{E}[\Delta s|f] > 0$. This motivates our reward function:

$$R_2(f_i) = \begin{cases} +1 & \text{if } s_{1,i} - s_0 > \delta \\ -1 & \text{if } s_{1,i} - s_0 < -\delta \\ 0 & \text{if } |s_{1,i} - s_0| \leq \delta \end{cases} \tag{29}$$

To understand why this reward structure is optimal, we analyze its expected value. Let $\Delta s_i = s_{1,i} - s_0$ and define the improvement distribution as $P(\Delta s)$. Then:

$$\mathbb{E}[R_2(f_i)] = \int_{\delta}^{\infty} P(\Delta s)\, d(\Delta s) - \int_{-\infty}^{-\delta} P(\Delta s)\, d(\Delta s) \tag{30}$$

$$= P(\Delta s > \delta) - P(\Delta s < -\delta) \tag{31}$$

For this expectation to be positive, we need $P(\Delta s > \delta) > P(\Delta s < -\delta)$, which occurs when the feedback provider generates more improvements than degradations. The policy gradient becomes:

$$\nabla_\theta \mathbb{E}[R_2(f_i)] = \mathbb{E}[R_2(f_i) \nabla_\theta \log \pi_\theta(f_i|s_0, a_0, x)] \tag{32}$$

Substituting our reward structure:

$$\nabla_\theta \mathbb{E}[R_2(f_i)] = \int_{\delta}^{\infty} P(\Delta s) \nabla_\theta \log \pi_\theta(f_i|s_0, a_0, x)\, d(\Delta s) \tag{33}$$

$$- \int_{-\infty}^{-\delta} P(\Delta s) \nabla_\theta \log \pi_\theta(f_i|s_0, a_0, x)\, d(\Delta s) \tag{34}$$

This gradient directly increases the probability of generating feedback that leads to score improvements while decreasing the probability of feedback that causes degradations.

However, a critical challenge emerges from the circular dependency between the feedback provider F_θ and scorer S_ϕ. If both components update simultaneously, we encounter a moving target problem. To formalize this instability, consider the joint dynamics of the two components.

At step t, the feedback provider parameters $\theta^{(t)}$ are updated based on rewards computed using scorer parameters $\phi^{(t)}$. Simultaneously, if the scorer updates to $\phi^{(t+1)}$, it changes the reward landscape for the next iteration. This creates a coupled dynamical system:

$$\theta^{(t+1)} = \theta^{(t)} + \alpha_\theta \nabla_\theta \mathbb{E}[R_2(f_i; \phi^{(t)})] \tag{35}$$

$$\phi^{(t+1)} = \phi^{(t)} + \alpha_\phi \nabla_\phi \mathbb{E}[R_1(r_j; \theta^{(t)})] \tag{36}$$

The problem is that the reward function R_2 depends on ϕ, so when ϕ changes, the reward signal for the same feedback changes, creating instability. To analyze this mathematically, consider the Jacobian of the combined system:

$$J = \begin{bmatrix} \frac{\partial}{\partial \theta} \nabla_\theta \mathbb{E}[R_2] & \frac{\partial}{\partial \phi} \nabla_\theta \mathbb{E}[R_2] \\ \frac{\partial}{\partial \theta} \nabla_\phi \mathbb{E}[R_1] & \frac{\partial}{\partial \phi} \nabla_\phi \mathbb{E}[R_1] \end{bmatrix} \tag{37}$$

The off-diagonal terms $\frac{\partial}{\partial \phi} \nabla_\theta \mathbb{E}[R_2]$ and $\frac{\partial}{\partial \theta} \nabla_\phi \mathbb{E}[R_1]$ represent the coupling between the two optimization problems. When these terms are large, the system can exhibit oscillatory or unstable behavior.

The momentum encoder strategy addresses this by decoupling the update frequencies. Instead of updating both parameters every step, we maintain:

$$\phi^{(t+1)} = \begin{cases} \phi^{(t)} & \text{if } t \bmod m \neq 0 \\ \theta^{(t)} & \text{if } t \bmod m = 0 \end{cases} \tag{38}$$

This creates periods of stability where ϕ remains fixed while θ optimizes against a consistent reward signal. During these intervals, the feedback provider's optimization problem becomes:

$$\max_{\theta} \mathbb{E}[R_2(f_i; \phi_{\text{fixed}})] \tag{39}$$

Since ϕ is fixed, the off-diagonal coupling terms vanish, and the optimization becomes stable. The convergence analysis during each fixed-ϕ period follows standard policy gradient theory.

When ϕ updates (every m steps), it incorporates the improved feedback generation capabilities developed during the previous period. This creates a staircase-like improvement pattern where each plateau represents stable optimization followed by a knowledge transfer step.

To analyze convergence, we need to distinguish between two different measures of performance. Let $Q(\theta)$ represent the true quality of feedback generated by parameters θ, measured by an idealized, consistent evaluation standard. In contrast, $S_\phi(\cdot)$ represents the score assigned by the current model parameters ϕ, which may vary across different parameter settings.

The crucial insight is that while $\theta^{(tm)}$ was optimized to maximize $\mathbb{E}[S_{\phi^{((t-1)m)}}(\tau_{\text{revised}}(\theta))]$, this does not guarantee that $\mathbb{E}[S_{\phi^{(tm)}}(\tau_{\text{revised}}(\theta^{(tm)}))] \geq \mathbb{E}[S_{\phi^{((t-1)m)}}(\tau_{\text{revised}}(\theta^{(tm)}))]$ because the scoring function itself has changed.

However, we can establish convergence through a different approach. Consider the sequence of feedback quality improvements measured by a fixed, external evaluation standard $Q^*(\cdot)$. During each interval $[(t-1)m, tm)$, the feedback provider θ is optimized according to:

$$\theta^{(k+1)} = \theta^{(k)} + \alpha \nabla_\theta \mathbb{E}[R_2(f; \phi^{((t-1)m)})] \tag{40}$$

Since the reward R_2 is designed to correlate with true improvement (i.e., $\mathbb{E}[R_2(f; \phi)] > 0$ when $Q^*(f) > Q^*(\text{baseline})$), we have:

$$Q^*(\theta^{(tm)}) \geq Q^*(\theta^{((t-1)m)}) \tag{41}$$

This inequality holds because $\theta^{(tm)}$ was specifically trained to generate feedback that leads to improvements as measured by a scorer that was previously optimized for the same objective.

The momentum update ensures that the new scorer $S_{\theta^{(tm)}}$ inherits the improved capabilities from the feedback training process. While we cannot guarantee that the numerical scores will increase, we can establish that the overall system capability improves monotonically.

To formalize this, define the system-wide performance as:

$$\Phi(t) = \max_{\tau} \mathbb{E}[Q^*(\tau) | \tau = P(F_{\theta^{(t)}}(\tau_0), \tau_0)] \tag{42}$$

This represents the best possible proof that can be achieved by applying feedback from the current model. Under our training scheme:

$$\Phi(tm) \geq \Phi((t-1)m) \tag{43}$$

The momentum update preserves this monotonic improvement while providing training stability. The key insight is that even though individual score values may fluctuate due to changing evaluation criteria, the underlying capability to generate effective feedback improves consistently.

C Training Approach Discussion

The backbone of Proof-Verifier is Qwen3-8B. Our model is trained exclusively using RL without prior SFT, as we find that additional SFT training reduces the diversity of the exploration space during rollout generation while providing only marginal performance improvements. We identify two main advantages for score distribution of SFT through pre-experiments: (1) distilling knowledge from larger models (e.g., Qwen3-235B) to smaller models improves the robustness of evaluation score distributions, and (2) applying self-consistency filtering strategies to datasets generated by the model itself for self-training also enhances robustness. However, we find that these benefits can be naturally integrated into the RL training process by designing reward function that provids positive rewards only for responses that are both correct and group-consistent. Additionally, while SFT traditionally is used to establish output format, this can be achieved directly through the reward function design. Thus, PROOF-VERIFIER is trained using GRPO entirely, with the reward function serving as the core design mechanism. The training process consists of two steps, detailed below, following a coarse-grained to fine-grained training objective progression.

C.1 EVALUATION STRATEGY DESIGN

When performing multiple evaluations on a single proof candidate, the stability and reliability of evaluation scores are crucial for robust assessment. We analyze two key factors that influence evaluation quality: (1) sampling strategy, including Single Sample, Parallel Sampling, and IID sampling, where the key distinction between the latter two is that parallel sampling generates trajectories that influence each other through various parameters (e.g., repetition penalties, beam group sizes), while IID sampling generates completely independent evaluations, and (2) prompt complexity, comparing simple prompts versus detailed rubric-based prompts. We conduct experiments across multiple benchmarks including CombiBench, FIMO, miniF2F, ProofNet, ProverBench, and PutnamBench. Using LLM-based provers, we first generate informal and formal proof candidates via IID sampling, then evaluate them using two models of different scales: Qwen3-8B and Qwen3-235B. Results are shown in Table 3, where outliers represent abnormal scores exceeding the expected [0-100] range, ICC measures the consistency of ratings across multiple evaluations of the same proof, and Median CV quantifies the relative variability in evaluation scores. We find that parallel sampling produces more stable evaluations than IID sampling, with lower outlier rates and higher consistency metrics. Additionally, parallel sampling achieves higher computational efficiency under VLLM-optimized infrastructure. Regarding prompt design, complex prompts containing detailed scoring rubrics surprisingly underperform compared to simple, direct prompts, likely due to increased instruction complexity leading to inconsistent interpretation. Therefore, we adopt Simple Prompt + Parallel Sampling for all subsequent experiments.

Sampling	**Median CV** $\downarrow$	**ICC** $\uparrow$	**Outlier(%)** $\downarrow$
	Qwen3-8B		
C+Parallel	0.1031	0.5540	4.3
$\hookrightarrow$ IID	0.1056	0.5537	4.7
S+Parallel	<u>0.0979</u>	<u>0.6140</u>	<u>1.5</u>
$\hookrightarrow$ IID	0.0983	0.5737	1.6
	Qwen3-235B		
C+Parallel	0.0770	0.8085	1.4
$\hookrightarrow$ IID	0.0769	0.8081	1.6
S+Parallel	<u>**0.0713**</u>	<u>**0.8372**</u>	<u>**0.2**</u>
$\hookrightarrow$ IID	0.0753	0.8128	0.3

Table 3: Consistency and reliability of evaluation scores across different models, prompts, and sampling strategies. S/C denotes simple/complex prompts, respectively.

D HUMAN ANNOTATION INTERFACE

We developed a web-based interface for human evaluation of mathematical proof assistance quality through pairwise comparisons within each response type.

D.1 INTERFACE DESIGN

The interface displays four components (Figure 7): the original mathematical statement at the top, the student's proof attempt below it, two response panels side-by-side in the center, and selection buttons at the bottom. Each comparison presents two responses of the same type (e.g., error analysis A vs. error analysis B).

D.2 EVALUATION SETUP

For each response type, annotators complete 20 pairwise comparisons. Panel positions are randomized to avoid bias. Annotators select which response would be more helpful for a student.

Figure 7: Human annotation interface for pairwise comparison of mathematical proof assistance types. The web-based evaluation tool presents annotators with a mathematical statement and two alternative response types from three categories: error analysis, actionable feedback, and refined proof.

D.3 WIN RATE ANALYSIS

After collecting preferences across all pairs, we analyze correlations between win rates within the same verification trajectory. For example, if verification trajectory A's error analysis outperforms trajectory B's error analysis, we examine whether trajectory A's actionable feedback also outperforms trajectory B's feedback, revealing consistency across response types.

E RELATED WORK

Mathematical theorem proving has seen significant progress with benchmarks like miniF2F (Zheng et al., 2022), FIMO (Liu et al., 2023), and PutnamBench (Tsoukalas et al., 2024), alongside advances in AI-assisted approaches such as AlphaGeometry (Trinh et al., 2024; Chervonyi et al., 2025) and recent neural theorem provers (Polu & Sutskever, 2020). Current verification relies primarily on interactive theorem provers like Lean4 (de Moura et al., 2015; de Moura & Ullrich, 2021), Coq (Huet & Paulin-Mohring, 2000), and Isabelle (Nipkow et al., 2002), with recent improvements in compilation efficiency through Kimina Lean Server (Santos et al., 2025). However, these approaches have critical limitations for RLVR applications: they only support formal languages while excluding natural language proofs, and provide only binary verification results without the detailed error analysis or actionable feedback essential for effective reinforcement learning.

Reward modeling for mathematical reasoning has primarily focused on traditional problem-solving tasks. Process reward models (Lightman et al., 2024; Wang et al., 2024a) have shown success in step-by-step verification for computational problems, while outcome reward models achieve strong results through final answer comparison (Cobbe et al., 2021). More broadly, LLM-as-a-judge approaches (Zheng et al., 2023) have evolved from instruction-following evaluation to reward models across domains including medical QA (Croxford et al., 2025), multimodal tasks (Chen et al., 2024), and code generation (Zhao et al., 2024), with open-source alternatives like PandaLM (Wang et al., 2024b) and Prometheus (Kim et al., 2024a;b) providing cost-effective solutions. While these methods succeed in traditional mathematical tasks through reference answer comparison (Chen et al., 2025), theorem proving's process-oriented nature and diverse solution paths present unique verification challenges that limit RLVR applications in this domain. Our work addresses this verification bottleneck by developing the first dual-language verifier specifically designed for mathematical theorem proving, enabling RLVR applications through detailed error analysis and fine-grained evaluation capabilities via novel two-stage reward modeling.

F TRAINING DATASET STATISTICS

Our initial data is divided into natural language data and formal language data, including the statement-proof pairs labeled to be correct or not.

For statement-proof pairs in natural language, we utilize the statements from OPC dateset Dekoninck et al. (2025) and RFM dataset Guo et al. (2025). We use different reasoning models to generate multiple proofs for a single statement, which we then label as either correct or incorrect. Of the statement-proof pairs in our initial pool, 2,000 are from the RFM dataset (582 labeled as correct, 1,418 as incorrect) and 3,039 are from the OPC dataset (1,109 labeled as correct, 1,930 as incorrect). To balance the dataset, we remove a number of incorrect proofs. The final dataset consists of 1,164 entries from the RFM dataset and 2,218 from the OPC dataset, with an equal number of incorrect and correct proofs. We evenly extract data from the formal statement-proof pairs of datasets including MiniF2F Zheng et al. (2022), ProofNet Azerbayev et al. (2023), PutnamBench Tsoukalas et al. (2024), ProverBench Ren et al. (2025), CombiBench Liu et al. (2025), Fimo Liu et al. (2023), and Hmmt Zhang et al. (2025b) (Note that the number of statements in these datasets varies, and the quantity of our dataset refers to the number of statement-proof pairs but not statements. Besides, dataset like Hmmt contains statements and proofs in natural language, and we translate them into formal language), and ensure that the total amount of formal data is roughly the same as that of natural language data. Ultimately, we obtain a final dataset of 6764 entries, as shown in Table 4. And we also count the distribution of knowledge domains, as shown in Table 5, which can be seen as a rough evaluation of the diversity of our dataset.

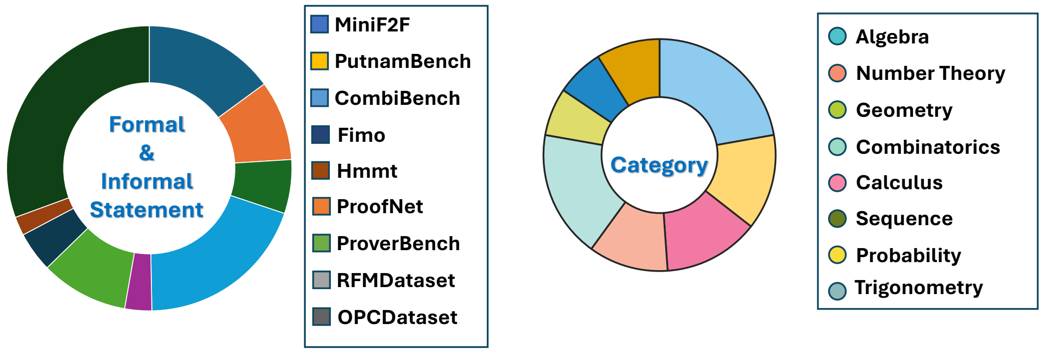


Figure 8: Benchmark statistics and category distribution.

Dataset Name	Data Size
OPC Dataset	2218
RFM Dataset	1164
MiniF2F	484
ProofNet	484
PutnamBench	484
ProverBench	484
CombiBench	484
Fimo	484
Hmmt	484
Total	6770

Table 4: Statistics of training set data sources.

G Prompt Template Design

These training datasets are formatted into a predefined prompt template for training. The evaluator is instructed to generate error analysis, evaluation scores, and actionable feedback as output. Natural language inputs include the statement and proof, while formal language inputs additionally include compilation results. We find this essential because without compilation results, the trained evaluator often produces incorrect analyses with low accuracy. This occurs because Lean4 proofs can encompass numerous proof steps and strategies within just one or two lines of code, requiring significant compilation time even for the compiler. Rather than having the evaluator perform lengthy reasoning to interpret the code and analyze potential errors, we directly provide compilation results to help it better understand the code and proof states, enabling more accurate and targeted feedback.

We compared two types of prompts: 1) simple free-style prompts that only constrain the format of model outputs, versus 2) complex prompts with specific guidance, such as requiring analysis from given perspectives, summing scores across multiple dimensions, and generating feedback from predefined frameworks. These two types of prompts are shown in Appendix M. Counterintuitively, we found that the first prompt yields significantly better results than the second. Even large models may make errors when summing evaluation scores and struggle to follow complex instructions. For the second type, models fail to follow the instruction to provide a detailed and in-depth analysis for each required perspective, instead offering only superficial and cursory responses. In contrast, simple prompts allow models to perform targeted, self-adaptive detailed analysis and provide specific feedback. For training, overly specific and complex prompts can be viewed as parameterized prefixes that constrain the exploration space of conditional generation rollouts, thereby preventing model improvement. Simple prompts provide models with sufficient search space, where as long as the format is correct, the optimization of intermediate processes relies on self-exploration, which

Domain	Data Size
Algebra	1480
Number Theory	1184
Calculus	958
Geometry	886
Combinatorics	740
Sequence	592
Probability	392
Trigonometry	422

Table 5: Statistics of different mathematical domains.

can fully leverage the strengths of RL to train more powerful models. Our experiments confirm this point: in RL, specific and complex prompts not only perform poorly initially but also provide limited improvement for the model. Detailed experimental results are shown in Appendix K.

H RFM DATASET

The prompt template follows a rubric-based evaluation containing 10 perspectives, covering various common proof errors such as Transformation Error, Hidden Assumption, and Boundary Neglect, as detailed in Appendix M. We employed three PhD-level math students to independently label the data (discarding entries where consensus was not reached), and comparison revealed a Pearson correlation coefficient of 0.89, thus demonstrating the feasibility of using this approach for large-scale labeling of this dataset, where the noise level remains within acceptable bounds.

1) OPC Dataset Dekoninck et al. (2025): Contains proof attempts generated by multiple advanced reasoning models using natural language on PutnamBench (Tsoukalas et al., 2024) statements. Multiple human experts label these proof attempts as True or False. For cases where annotators disagree, we apply majority voting when feasible; otherwise, we discard instances with substantial disagreement.

2) RFM Bench Guo et al. (2025): A benchmark contains proof statements across multiple difficulty levels (high school, undergraduate, graduate) and mathematical domains, including geometry, algebra, inequalities, number theory, and calculus. All statements are human-annotated and verified for quality and difficulty control. Since this dataset contains only statements, we generate proof candidates using advanced reasoning models and employ LLM-as-a-judge labeling with Gemini-2.5-Pro as the evaluator, as we find that Gemini-2.5-Pro demonstrates strong proof evaluation capabilities on this dataset under carefully designed rubrics. Finally, we extract 100 {statement, proof attempt} pairs for human annotation to form part of our test set.

Both datasets may introduce validation errors, potentially overlooking logical issues and other subtle errors that are difficult to detect. Moreover, they contain only natural language proofs without formal language coverage. To address these limitations, we introduce a third data construction method.

I TEST DATASET COLLECTION

For formal language, we selected different prover models (DeepSeek Prover V2 Ren et al. (2025), Kimina Prover Wang et al. (2025a), Goedel Prover Lin et al. (2025), Leanabell Prover Zhang et al. (2025a), and STP Prover Dong & Ma (2025)) that differ from training data collection models, sampling 32 proof attempts per statement on new benchmarks with compiler-generated labels. For natural language, we used various series models (Qwen3, Mistral, Magistral Mistral-AI et al. (2025), Qwen 2.5, Gemma Team et al. (2025), DeepSeek R1) for sampling, with 100 samples labeled by human annotators and the remaining labeled by Gemini-2.5-Pro. Our three human annotators achieved Cohen's Kappa Cohen (1960) scores of 0.86, 0.86, and 0.88, validating the high quality and reliability of the test set. The final test set contains 5k formal language and 1k natural language samples.

J MODEL INTRODUCTION

J.1 GENERAL PROPOSE MODELS

Qwen series. We use the Qwen-2.5-7B, Qwen-2.5-72B, Qwen-3-8B and Qwen3-30B Qwen et al. (2025) in our experiments. Both Qwen2.5-8B and Qwen3-8B acquire extensive mathematical knowledge during pre-training, and Qwen3-8B, in particular, has ability to handle more difficult mathematical problems. As a result, they are widely used as base models for training in research. Qwen-2.5-72B is the largest open-source model in Qwen-2.5 series, which demonstrates significant improvements in mathematical ability compared to its predecessors. Qwen3-30B, a new representative model of the Qwen series, adopts a mixture-of-experts architecture and can employ long chain-of-thoughts, which greatly improves its mathematical reasoning capability. Specifically, the Qwen3-30B model shows strong performance on competition-level math benchmarks, such as AIME-2024 Jia (2025) and AIME-2025 OpenCompass Team (2025).

Mistral and Magistral series. Mistral Mistral-AI et al. (2025) is a series of powerful, efficient language models from the Mistral AI. The models are known for their strong performance across various tasks while being efficient. These models achieve good performance on various mathematical benchmarks like MATH Hendrycks et al. (2021) and GSM8K Cobbe et al. (2021), proving their strong capabilities from basic arithmetic to advanced problem-solving. To further improve models' reasoning ability, Mistral AI introduced the Magistral series. These new models are specifically trained for advanced reasoning tasks and capable of performing long chain-of-thought. This makes their problem-solving process more transparent and reliable for complex applications, while achieving high scores on challenging math benchmarks like AIME-2024 Jia (2025) and AIME-2025 OpenCompass Team (2025).

Gemma series. Gemma Team et al. (2025) is a family of lightweight, open-source models developed by Google. The recent Gemma-3 series represents a significant leap forward in capabilities and efficiency. A key highlight of the Gemma 3 series is its exceptional performance in mathematical reasoning. The technical report confirms that Gemma 3 models demonstrate substantial gains over their predecessors on critical math benchmarks, specifically MATH and HiddenMath.

Deepseek-R1. Deepseek-R1 DeepSeek-AI et al. (2025) is one of the most prominent open-source reasoning models, widely used for complex mathematical reasoning. Its performance on two competition-level datasets, AIME-2024 and AIME-2025, is comparable to top commercial models, achieving a pass@1 score of nearly 90% or higher. Furthermore, Deepseek-R1's recent performance on the RFM Dataset for mathematical proofs is shown to surpass many reasoning models.

Gemini-2.5-pro. The Gemini-2.5-pro Comanici et al. (2025) model is dedicated to pushing the frontier of AI with advanced reasoning. In addition to its excellent performance on common math competition datasets like AIME, this model has also recently been shown to surpass other models in its mathematical proof generation ability under the same criteria.

J.2 PROVER MODELS

In this paper, we refer to models that are specifically trained to generate formal proofs from formal statements as prover models. In this subsection, we provide a short introduction to the prover models we used.

DeepSeek-Prover-V2. DeepSeek-Prover-V2 Ren et al. (2025) is an open source prover model for theorem proving in Lean 4, which is distinguished by its innovative pipeline that effectively unifies informal, human-like reasoning with the rigor of formal proof generation. Its core feature is a two-stage process that begins by using a powerful general model (DeepSeek-V3 DeepSeek-AI et al. (2025)) to decompose complex theorems into a high-level proof sketch composed of simpler subgoals. These subgoals are then solved by a more specialized prover model.

KIMINA Prover. KIMINA Prover Wang et al. (2025a) emulates human problem solving by generating a mix of informal mathematical intuition and formal Lean 4 code. This process allows it to iteratively build and refine a proof.

Goedel Prover. The core innovation of Goedel Prover Lin et al. (2025) lies in addressing the scarcity of formal mathematical data through a two-pronged approach. It automatically translates a massive dataset of 1.64 million math problems from natural language into formal statements. Besides, it employs a training method where the model generates proofs for these statements, and any new correct proofs are added back into the training set to iteratively create a more powerful prover.

Leanabell Prover. Leanabell Prover Zhang et al. (2025a)undergoes a two-stage post-training strategy. The first stage is supervised fine-tuning on a large, custom-curated dataset. This dataset not only includes a massive collection of statement-proof pairs but, more importantly, incorporates synthetic data designed to integrate self-reflection and error correction. This is achieved by generating Chain-of-Thought style explanations for why a proof failed and how it is corrected. Second, using the GRPO algorithm, the finetuned model learns by generating entire proofs and receiving a direct reward signal from the Lean 4 compiler based on whether the proof is successfully verified. This strategy of combining SFT with cognitive data and RL optimization allows Leanabell-Prover to achieve good performance.

STP Prover. The training of STP prover Dong & Ma (2025) simultaneously operates in two roles of prover and conjecturer. These two roles create a dynamic self-play feedback loop. The conjecturer creates novel problems that are barely provable by the prover's current ability. This process generates a continuous stream of appropriately difficult training data. This allows the model to improve its reasoning skills without requiring more human-created datasets, effectively creating its own adaptive learning curriculum.

K Ablation Study on Prompt Design

Verifier	Acc	Prec	Rec	F1
Simple Prompt	0.57	0.58	0.55	0.56
Simple Prompt + RL	0.86	0.87	0.85	0.86
Complex Prompt	0.43	0.45	0.42	0.43
Complex Prompt + RL	0.52	0.53	0.51	0.52

Table 6: Comparison of prompt types during inference and after RL training. Simple Prompt outperforms Complex Prompt in both settings.

As shown in Table 6, we found that Simple Prompt performs better than Complex Prompt in both zero-shot stage and after RL training. The reason is that overly complex prompts are difficult for models to follow, resulting in poor inference performance. Additionally, overly complex prompts limit the model's exploration space, where model outputs become constrained and RL cannot effectively optimize the model further.

L Translation Error

In this section, we provide cases to show the errors that occurred when translating the formal statements or proofs to natural language statements or proofs.

L.1 Statement Transformation Error

We identify three main kinds of statement transformation errors: *Premise Difference*, *Conclusion Difference*, *Not to Prove*. *Premise Difference* refers to a discrepancy where the premise provided by the translated natural language statement differs from those off the original formal statement,as shown in Figure 9, 10, 11. In each figure, we mark the translation errors in red and provided an analysis of the errors.Taking Figure 9 as an example, the original formal statement does not provide aby specific value for the function f(x) and its corresponding inverse function at a given point, but only presented an equality condition. However, the translated natural language statement does include a value. Although these values do not affect the solution in this specific example, it still highlights a semantic inconsistency.

Conclusion Difference refers to a discrepancy where the conclusion to be proven in the formal language statement differs from the conclusion of the translated natural language statement, as shown in Figure 10, 11, 12, 13, 14. For example, in Figure 12, the conclusion to be proven in the formal language statement is that a certain property holds for all positive integers. However, the natural language statement is incorrectly rendered as proving there are infinitely many such integers. While these two concepts are somewhat related, the difference between them is significant.

Not to Prove refers to a situation where the natural language statement fails to clearly distinguish between the premises and the conclusion to be proven(Figure 15, 16). In some cases, the translation is even a problem that requires a numerical result instead of a proof(Figure 17).

Formal Statement and proof:

```
import Mathlib
import Aesop
set_option maxHeartbeats 0
open BigOperators Real Nat Topology Rat
/-- Suppose f(x) is an invertible function with f(2)=f^{-1}(2)=4.
    Show f(f(2)) = 2. -/
theorem mathd_algebra_188 (σ :  Equiv R R)
    (h :  σ⁻¹ 2 = σ 2) :
    σ (σ 2) = 2 := by
```

Transformed Natural Language Statement:

Suppose f is a bijective function from the real numbers to the real numbers, with the property that both $f(2)$ and its inverse function $f^{-1}(2)$ evaluate to 4. What is the value of $f(f(2))$? Show that it equals 2.

Error Type: Premise Difference
Error Analysis: The natural language statement describes the specific values $f(2) = 4$ and $f^{-1}(2) = 4$, while the Lean code states that $f(2)$ and $f^{-1}(2)$ are equal, without explicitly mentioning the value 4. This error may have been affected by the comment content in the lean code.

Figure 9: A failure case of *Premise Difference*.

L.2 Proof Translation Error

We also explored the bad cases of converting formal proofs into natural language proofs. A common case is that the natural language proof translated by the model is merely an annotation of the formal proof, as shown in Figure 18 and 19. This means it is not a complete proof on its own, because languages like Lean 4 contain elements such as tactics that are not fully explained in the translation. Furthermore, we also discover cases where the natural language proofs and formal proofs use different methods(Figure 20). Although these proofs may be correct on their own, they fail to meet our consistency requirement.

M Prompts

In this section, we detail the prompts used in our experiments.

Figure 21 is the prompt we provide to *Gemini-2.5-pro* to judge the proofs generated by various LLMs for statements from RFM Dataset. This prompt not only asks the model to provide an overall correctness score, but also requires it to categorize any errors into one or more failure modes, which are detailed in Figure 22.

Figure 23 and 24 show the prompts provided to our evaluator to evaluate a formalized proof, with and without verification information from the Lean 4 compiler, respectively.

Figure 25 and 27 show the prompts provided to our evaluator to evaluate a natural language proof, with and without detailed rubrics to give the final score, respectively.

Formal Statement and proof:

```
import Mathlib
open Matrix
def coeff_matrix : Matrix (Fin 4) (Fin 3) ℚ :=
  !![![ -19,  8,  0],
      ![ -71, 30,  0],
      ![ -2,   1,  0],
      ![  4,   0,  1]]
def aug_matrix : Matrix (Fin 4) (Fin 4) ℚ :=
  !![![ -19,  8,  0, -108],
      ![ -71, 30,  0, -404],
      ![ -2,   1,  0, -12],
      ![  4,   0,  1,  14]]
axiom rank_of_matrix :  rank coeff_matrix = 2 ∧ rank aug_matrix = 2
axiom verify_solution :  (4, -4, -2) ∈ {x :  ℚ × ℚ × ℚ |
-19 * x.1 + 8 * x.2.1 = -108 ∧
-71 * x.1 + 30 * x.2.1 = -404 ∧
-2 * x.1 + x.2.1 = -12 ∧
4 * x.1 + x.2.2 = 14}
/-- The system of linear equations has a unique solution (4, -4, -2).
    -/
theorem unique_solution : ∃! x : ℚ × ℚ × ℚ,
  -19 * x.1 + 8 * x.2.1 = -108 ∧
  -71 * x.1 + 30 * x.2.1 = -404 ∧
  -2 * x.1 + x.2.1 = -12 ∧
  4 * x.1 + x.2.2 = 14 := by
```

Transformed Natural Language Statement:

Consider the system of linear equations:
1) -19x + 8y = -108
2) -71x + 30y = -404
3) -2x + y = -12
4) 4x + z = 14
Prove that this system has a unique solution (4, -4, -2).

Error Type: Premise Difference and Conclusion Difference
Error Analysis: In the formalized statement, it is directly given that the rank of the coefficient matrix is equal to the rank of the augmented matrix, which falls under the category of premise difference. The formalized statement directly indicates that (4, -4, 2) is a solution to the equation, so it only needs to prove uniqueness. However, the natural language statement first needs to prove that (4, -4, 2) is a solution to the equation, which belongs to conclusion difference.

Figure 10: A failure case of *Premise Difference* and *Conclusion Difference*.

N Feedback Case Study

To intuitively demonstrate the effectiveness of our evaluator, we present some cases in this section, as shown in Figures 28, 29, 30, and 31. Figures 28, 29, and 30 show feedback on formal proofs, while Figure 31 provides feedback on a natural language proof.

From the feedback on formal proofs, we can clearly see that the evaluator's feedback can identify and correct errors related to the properties of Lean, as well as point out the missing parts in the formal proof. For example, the case shown in Figure 30 indicated that the feedback precisely identifies the critical gap, which is the missing proof that $\pi < 4$ required to apply the function's conditional definition—and resolves it by directing the user to the essential Mathlib theorem Real.pi_lt_four. It corrects the ineffective simp_all tactic by prescribing explicit hypothesis applica-

Formal Statement and proof:

```
import Mathlib
/-- Let f(u,v) be differentiable with
    ∂f/∂u − ∂f/∂v = 2(u − v)e^{−(u+v)}
    and f(u,0) = u^2 e^{−u}. Then:
    a) For g(x,y) = f(x,y − x) we have ∂g/∂x = (4x − 2y)e^{−y};
    b) f(u,v) = (u^2 + v^2)e^{−(u+v)};
    c) f has a unique global minimum at (0,0) with f(0,0) = 0. -/
open Real
variable {u v x y :  R }
noncomputable def f (u v :  R) : R) := (u^2 + v^2) * exp (-(u + v))
noncomputable def g (x y :  R) : R) := f x (y - x)
axiom f_partial_condition (u v :  R) :
deriv (fun u => f u v) u - deriv (fun v => f u v) v = 2 * (u - v) *
    exp (-(u+v))
axiom f_initial_condition (u :  R)) :
f u 0 = u^2 * exp (-u)
axiom partial_g_x (x y :  R) :
deriv (fun x => g x y) x = (4 * x - 2 * y) * exp (-y)
lemma g_expression (x y :  R) :
g x y = (x^2 + (y - x)^2) * exp (-y) :=
```

Transformed Natural Language Statement:

Consider a differentiable function $f(u, v)$ that satisfies:
1. The difference between its partial derivative with respect to u and its partial derivative with respect to v equals $2(u - v)e^{-(u+v)}$
2. The initial condition $f(u, 0) = u^2 e^{-u}$

We are to show that:
a) For $g(x, y) = f(x, y - x)$, the partial derivative of g with respect to x is $(4x - 2y)e^{-y}$
b) The function $f(u, v)$ has the explicit form $(u^2 + v^2)e^{-(u+v)}$
c) The function f has a unique global minimum at $(0, 0)$ with value 0

Error Type: Premise Difference and Conclusion Difference
Error Analysis: In the formalized statement, it is only necessary to prove that g(x, y) = (x^2 + (y - x)^2) * exp(-y). However, the natural language statement needs to prove three statements. Among them, statements (a) and (b) are given as known conditions in the formalized statement, and statement (c) is not reflected in the lean code. Therefore, it falls into both the categories of *Premise Difference* and *Conclusion Difference*.

Figure 11: A failure case of *Premise Difference and Conclusion Difference*

tion (h_1 _ ⟨Real.pi_nonneg, Real.pi_lt_four⟩), transforming an incomplete automation attempt into a robust, verifiable proof that adheres strictly to the function's domain conditions.

From the feedback on natural language proofs shown in Figure 31, we can see the feedback accurately point out that the inequality used in the proof is invalid and does not hold under certain conditions. It even provided failure cases of the inequality and a corresponding improvement plan.

Formal Statement and proof:

```
import Mathlib
import Aesop
set_option maxHeartbeats 0
open BigOperators Real Nat Topology Rat
/-- For how many positive integers m does there exist at least one
    positive integer n
such that m·n ≤ m+n?
(A) 4    (B) 6    (C) 9    (D)12    (E)infinitely
    many. Show that it is (E)infinitely many.-/
theorem amc12a_2002_p6 (n : ℕ)
     (h_0 :  0 < n) :
     ∃m, m > n ∧ ∃p, m * p ≤ m + p  := by
```

Transformed Natural Language Statement:

How many positive integers m exist such that there is at least one positive integer n satisfying the inequality $m \cdot n \leq m + n$? The options are: - (A) 4 - (B) 6 - (C) 9 - (D) 12 - (E) infinitely many.
We are to show that the correct answer is (E) infinitely many.

Error Type: Conclusion Difference
Error Analysis: The formalized statement means that a certain conclusion holds for any n greater than 0, while the natural language statement means that there are infinitely many values of n for which this conclusion holds. Therefore, this belongs to a conclusion difference.

Figure 12: A failure case of *Conclusion Difference*

Formal Statement and proof:

```
import Mathlib
/-- Let A be a real 2×2 matrix. Then the determinant of A^2 is always
     nonnegative, i.e., det(A^2) ≥ 0.-/
open Matrix
variable (A : Matrix (Fin 2) (Fin 2) ℝ)
lemma det_A_sq : det (A * A) = (det A) ^ 2 := by
```

Transformed Natural Language Statement:

Let A be a real 2×2 matrix. Show that the determinant of A^2 is always non-negative, i.e., $\det(A^2) \geq 0$.

Error Type: Conclusion Difference
Error Analysis: In the formalized statement, it is required to prove that the determinant of a matrix squared is equal to the square of the determinant. However, the natural language statement needs to prove that the determinant of a matrix squared is greater than or equal to 0. Therefore, this belongs to a Conclusion Difference.

Figure 13: A failure case of *Conclusion Difference*.

Formal Statement and proof:

```
import Mathlib
import Aesop
set_option maxHeartbeats 0
open BigOperators Real Nat Topology Rat

/-- What is the tens digit of 5^2005? Show that it is 2. -/
theorem mathd_numbertheory_198 :
  5 ^ 2005 % 100 = 25 := by
```

Transformed Natural Language Statement:

What is the tens digit of 5^{2005}? Prove that it is 2.

Error Type: Conclusion Difference
Error Analysis: The statement in the formal language requires proving that the remainder is 25, while the statement in natural language only needs to prove that the tens digit of the remainder is 2.

Figure 14: A failure case of *Conclusion Difference*.

Formal Statement and proof:

```
import Mathlib
/- The composition of the functions u(x) = sin x and v(x) = x^2 is
     u(v(x)) = sin(x^2). -/
open Real
noncomputable def u (x : R) : R := sin x
def v (x : R) : R := x^2
theorem composition_of_sine_and_quadratic (x : R) :
  u (v x) = sin(x^2) := by
```

Transformed Natural Language Statement:

What is the composition of the functions $u(x) = \sin x$ and $v(x) = x^2$, and how does it simplify? The composition $u(v(x))$ simplifies to:

$$u(v(x)) = \sin(x^2)$$

Error Type: Not to Prove
Error Analysis: The natural language statement presents the content of the formal language as a factual elaboration, rather than treating it as a mathematical proof problem.

Figure 15: A failure case of *Not to Prove*.

Formal Statement and proof:

```
import Mathlib
import Aesop
set_option maxHeartbeats 0
open BigOperators Real Nat Topology Rat

/- Let f(x) = x^3 - 9x^2 + 24x be a real-valued function defined on ℝ. Then
    :
    1. f(x) has a local maximum at x = 2.
    2. f(x) has a local minimum at x = 4.
    3. These are the only local extrema of f(x) on ℝ. -/
open Real

noncomputable def f (x : ℝ) : ℝ := x^3 - 9 * x^2 + 24 * x

noncomputable def f_deriv (x : ℝ) : ℝ := 3 * x^2 - 18 * x + 24

noncomputable def f_second_deriv (x : ℝ) : ℝ := 6 * x - 18

axiom critical_points : f_deriv x = 0 ↔ x = 2 ∨ x = 4

lemma second_derivative_test (x : ℝ) (h : f_deriv x = 0) :
  ((x = 2 ∧ f_second_deriv x < 0) ∨ (x = 4 ∧ f_second_deriv x > 0))
     := by
```

Transformed Natural Language Statement:

Consider the real-valued function $f(x) = x^3 - 9x^2 + 24x$ defined on all real numbers. The statement claims that: 1. The function has a local maximum at $x = 2$ 2. The function has a local minimum at $x = 4$ 3. These are the only local extrema of the function on the real numbers

Error Type: Not to Prove
Error Analysis: The natural language statement expounds on the claim of the formal language, yet it is not a mathematical proof problem itself.

Figure 16: A failure case of *Not to Prove*.

Formal Statement and proof:

```
import Mathlib
import Aesop
set_option maxHeartbeats 0
open BigOperators Real Nat Topology Rat

/-- Solve the system:
    x + y + z = a
    x^2 + y^2 + z^2 = b^2
    xy = z^2
    where a and b are constants.
    Give the conditions on a and b so that x, y, z are distinct
        positive numbers. -/
theorem imo_1961_p1 (x y z a b : ℝ)
  (h_0 : 0 < x ∧ 0 < y ∧ 0 < z)
  (h_1 : x ≠ y) (h_2 : y ≠ z) (h_3 : z ≠ x)
  (h_4 : x + y + z = a)
  (h_5 : x^2 + y^2 + z^2 = b^2)
  (h_6 : x * y = z^2) :
  0 < a ∧ b^2 < z^2 ∧ a^2 < 3 * b^2 := by
```

Transformed Natural Language Statement:

Given three positive real numbers x, y, z that are all distinct, and two constants a and b, solve the system of equations:
1. $x + y + z = a$
2. $x^2 + y^2 + z^2 = b^2$
3. $xy = z^2$
Find the conditions that a and b must satisfy for the system to have distinct positive real solutions x, y, z.

Error Type: Not to Prove
Error Analysis: The natural language statement requires finding the conditions that meet the problem, rather than proving the problem described by the formal language.

Figure 17: A failure case of *Not to Prove*.

Formal Statement and proof:

```
import Mathlib
open Real
variable {u v x y : ℝ}
noncomputable def f (u v : ℝ) : ℝ := (u^2 + v^2) * exp (-(u + v))
/-- The function g defined in terms of f -/
noncomputable def g (x y : ℝ) : ℝ := f x (y - x)
axiom f_partial_condition (u v : ℝ) :
  deriv (fun u => f u v) u - deriv (fun v => f u v) v = 2 * (u - v) *
       exp (-(u+v))
axiom f_initial_condition (u : ℝ) :
  f u 0 = u^2 * exp (-u)
axiom partial_g_x (x y : ℝ) :
  deriv (fun x => g x y) x = (4 * x - 2 * y) * exp (-y)
lemma g_expression (x y : ℝ) :
  g x y = (x^2 + (y - x)^2) * exp (-y) := by
  simp only [g, f]
  ring
```

Transformed Natural Language Proof:

The proof proceeds as follows:
1. We start with the definition of $g(x, y)$ as $f(x, y - x)$, where f is defined to be $(u^2 + v^2)e^{-(u+v)}$.
2. The goal is to show that $g(x, y)$ equals $(x^2 + (y - x)^2)e^{-y}$.
3. By substituting the definition of f into g, we get: $g(x, y) = f(x, y - x) = (x^2 + (y - x)^2)e^{-(x+(y-x))}$
4. Simplifying the exponent: $-(x + (y - x)) = -y$
5. Thus we have $g(x, y) = (x^2 + (y - x)^2)e^{-y}$
6. The proof uses the 'simp' tactic in Lean to simplify the expression, focusing only on the definitions of g and f.
7. The 'ring' tactic is then used to verify that the algebraic manipulations are correct.
This shows that the given expression for $g(x, y)$ in terms of f is indeed correct, which supports part (b) of the statement about the form of $f(u, v)$.

Error Analysis: This natural language proof is actually an explanation of the formal language proof and cannot exist as an independent solution to the problem. Especially in the key parts of the problem, it only explains the tactics used in the formal language without providing a correct natural language proof.

Figure 18: A failure case of translating formal proof to natural language proof.

Formal Statement and proof:

```
import Mathlib
import Aesop
set_option maxHeartbeats 0
open BigOperators Real Nat Topology Rat
/-- If \( Q = 11-5i \), \( E = 11+5i \), and \( D = 2i \), find \( Q\
    cdot E \cdot D \). Show that it is 292i.-/
theorem mathd_algebra_192 (q e d : \mathbb{C})
(h_0 : q = 11 - 5 * Complex.I) (h_1 : e = 11 + 5 * Complex.I)
(h_2 : d = 2 * Complex.I) : q * e * d = 292 * Complex.I := by
rw [h_0, h_1, h_2]
simp only [Complex.ext_iff, Complex.mul_re,
Complex.mul_im, Complex.add_re,
Complex.add_im,
Complex.ofReal_re, Complex.ofReal_im,
Complex.ofReal_add, Complex.ofReal_mul,
Complex.I_re,
Complex.I_im]
norm_num
<;> ring
<;> norm_num
<;> ring
```

Transformed Natural Language Proof:

To prove that $Q \times E \times D = 292i$, we follow these steps:
1. First, we substitute the given values into the expression: - Replace Q with $(11 - 5i)$ - Replace E with $(11 + 5i)$ - Replace D with $(2i)$
2. We then expand the multiplication step by step: - First multiply Q and E using complex multiplication rules - Then multiply the result by D
3. During the expansion, we: - Separate the real and imaginary parts of the complex numbers - Apply the distributive property of multiplication - Simplify using the fact that $i^2 = -1$ - Combine like terms
4. The simplification process involves: - Calculating the real parts (which should cancel out) - Calculating the imaginary parts (which should combine to 292) - Using basic arithmetic operations (addition, multiplication) - Normalizing the numerical expressions
5. After all simplifications, we verify that: - The real part becomes 0 - The imaginary part becomes exactly 292 - Thus the final result is indeed $292i$

Error Analysis: The proof in formal language uses Lean's internal algebraic simplification mechanism, and its correctness can be verified. However, the corresponding natural language proof must provide a detailed simplification process, which the above proof fails to do.

Figure 19: A failure case of ranslating formal proof to natural language proof.

Formal Statement and proof:

```
import Mathlib
import Aesop
set_option maxHeartbeats 0
open BigOperators Real Nat Topology Rat

/-- Show that there are infinitely many positive integers m for
    which
      there exists at least one positive integer n such that
         m·n ≤ m+n. -/
theorem amc12a_2002_p6 (n : N) (h_0 : 0 < n) :
  ∃ m, m > n ∧ ∃ p, m * p ≤ m + p := by
  /- For any positive n, choose m = n+1 and p = 1.
     Then (n+1)*1 = n+1 ≤ n+2 = (n+1)+1, so the inequality holds. -/
  use n + 1
  constructor
  · exact Nat.lt_succ_self n        -- n+1 > n
  · use 1
    simp                            -- (n+1)*1 ≤ (n+1)+1
```

Transformed Natural Language Proof:

To prove that there are infinitely many positive integers m for which there exists at least one positive integer n such that $m \cdot n \leq m + n$, we proceed as follows:
1. Constructing m: For any given positive integer n, define $m = n + 1$. This ensures $m > n$ (since $n + 1 > n$).
2. Verifying the inequality: Substitute $m = n + 1$ into $m \cdot n \leq m + n$:

$$(n + 1) \cdot n \leq (n + 1) + n$$

Simplify: $n^2 + n \leq 2n + 1$. For Lean's proof, choosing $p = 1$ simplifies the inequality to:

$$(n + 1) \cdot 1 \leq (n + 1) + 1 \implies n + 1 \leq n + 2$$

which is always true.
3. Conclusion: For every n, $m = n + 1$ works with $p = 1$, so there are infinitely many such m. The answer is (E) infinitely many.
Note: Lean uses $p = 1$ for simplification, but the key idea of infinitely many m remains.

Error Analysis: The formal language proof and the natural language proof adopt different approaches.

Figure 20: A failure case of translating formal proof to natural language proof.

Evaluation Prompt(RFM Dataset):

You are a math teacher highly skilled in evaluating proofs.
Problem Statement: [*Original Question*]
Student's Answer: [*LLM generated Proof*]

Your Tasks
1. **Detailed Analysis**: Read the student's answer and thought process carefully.
- If a claim is obvious in students's answer, you may accept it as valid provided the overall logic is sound.
- If the category **"Others"** is triggered, briefly describe the error in the Analysis section.

2. **Binary Scoring**: After your analysis, output *only* the exact block below, inside triple back-ticks, nothing else:
“
Error Pattern Analysis
- Transformation Error: 1|0
- Over Generalization: 1|0
- Invalid Construction: 1|0
- Wrong Division: 1|0
- Circular Reasoning: 1|0
- Logic Violation: 1|0
- Hidden Assumption: 1|0
- Boundary Neglect: 1|0
- Vague Argument: 1|0
- Vague Argument: 1|0
- Others: 1|0

Overall Correctness
- 1|0
”

Error Pattern Rubric
Presented in Figure 22

Scoring Semantics
- In **Error Pattern Analysis**: "1" = this error pattern **is present**. "0" = this error pattern **is NOT present**.
- In **Overall Correctness**: "1" = the proof is **completely correct** (no errors). "0" = the proof **contains at least one error**.

Consistency Rule
If **any** error pattern is "1", then **Overall Correctness must be "0"**. Only when **all** error patterns are "0" is Overall Correctness "1".

Do not output anything after the code block. Your answer is:

Figure 21: The evaluation prompt we use when assessing the answers to questions in the RFM Dataset generated by various models, which is provided to the *Gemini-2.5-pro* model.

Error Pattern Rubric:

1.
Category: **Transformation Error**
Definition: Recasting the target statement into a non-equivalent or strictly weaker one.
Typical example: To prove convergence of $\sum a_n$, only prove $\lim a_n = 0$; or replace "$A \iff B$" with "$A \Rightarrow B$".
2.
Category: **Over Generalization**
Definition: Inferring a universal claim from a few special or hand-picked cases.
Typical example: Verifying for $n = 1, 3, 5$ then claiming the result holds for all $n \in (N)$.
3.
Category: **Invalid Construction**
Definition: Failing to construct an object that should exist, or constructing one that doesn't meet requirements.
Typical example: Claiming a function that is everywhere linear yet nowhere differentiable.
4.
Category: **Wrong Division**
Definition: Partitioning into cases that miss at least one legitimate possibility or overlap.
Typical example: When analyzing the behavior of a function, dividing cases as "always positive," "always zero," and "always negative."
5.
Category: **Circular Reasoning**
Definition: Using the conclusion (or an equivalent reformulation) as a hidden or explicit premise.
Typical example: Assuming B when trying to prove $A \Rightarrow B$.
6.
Category: **Logic Violation**
Definition: A deduction step that contradicts logical or algebraic rules.
Typical example: From $a < b$ and $c < d$ concluding $a - c < b - d$ without checking signs.
7.
Category: **Hidden Assumption**
Definition: Applying a theorem or step whose hypotheses were neither stated nor proven.
Typical example: Differentiating a function known only to be continuous.
8.
Category: **Boundary Neglect**
Definition: Ignoring edge cases, endpoints, or limiting situations so the argument holds only "in the middle."
Typical example: Proving $f(x) = \sqrt(x)$ differentiable on $[0, 1]$ without checking at $x = 0$.
9.
Category: **Vague Argument**
Definition: Relying on intuition, diagrams, or "obvious" without formal justification.
Typical example: "The series obviously converges because the terms get smaller."
10.
Category: **Incomplete Proof**
Definition: mitting an essential component such as the converse, base case, or a logical bridge.
Typical example: Proving sufficiency but not necessity in an "if and only if."
11.
Category: **Others**
Definition: Any error not covered by the categories above.

Figure 22: The error pattern rubric used by the prompt shown in Figure 21.

Lean 4 Evaluation Template w/ Verification

<TASK_TYPE>
formal_proof_evaluation
<TASK_TYPE>

<THEOREM>
{theorem_statement}
<THEOREM>

<PROOF_ATTEMPT>
{proof_code}
<PROOF_ATTEMPT>

<VERIFICATION_RESULT>
{verification_output}
<VERIFICATION_RESULT>

Please evaluate this Lean 4 proof attempt and provide structured feedback.
Your response must follow this exact format:

<ERROR_ANALYSIS>
[Provide detailed technical analysis of the error, including error type classification, root cause, and severity assessment]
<ERROR_ANALYSIS>

<SCORE>
[Provide a numerical score from 0-100]
<SCORE>

<FEEDBACK>
[Provide specific, actionable suggestions for fixing the proof, including concrete code changes and alternative approaches]
<FEEDBACK>

Figure 23: The prompt with verification provided to evaluator to evaluate formal proof.

Lean 4 Evaluation Template w/o Verification

<**TASK_TYPE**>
formal_proof_evaluation
<**TASK_TYPE**>

<**THEOREM**>
{theorem_statement}
<**THEOREM**>

<**PROOF_ATTEMPT**>
{proof_code}
<**PROOF_ATTEMPT**>

Please evaluate this Lean 4 proof attempt and provide structured feedback.
Your response must follow this exact format:

<**ERROR_ANALYSIS**>
[Provide detailed technical analysis of the error, including error type classification, root cause, and severity assessment]
<**ERROR_ANALYSIS**>

<**SCORE**>
[Provide a numerical score from 0-100]
<**SCORE**>

<**FEEDBACK**>
[Provide specific, actionable suggestions for fixing the proof, including concrete code changes and alternative approaches]
<**FEEDBACK**>

Figure 24: The prompt without verification provided to evaluator to evaluate formal proof.

Natural Language Proof Evaluation Prompt w/ Rubrics

<**TASK_TYPE**>
natural_proof_evaluation
<**TASK_TYPE**>
<**PROBLEM**>
{problem_statement}
<**PROBLEM**>
<**PROOF_ATTEMPT**>
{proof_text}
<**PROOF_ATTEMPT**>
Please evaluate this natural language mathematical proof from the folowing rubrics and provide structured feedback.
<**RUBRICS**>
<Rubrics> <**RUBRICS**>
Your response must follow this exact format:
<**ERROR_ANALYSIS**>
[Follow the above rubrics to provide a detailed conceptual analysis step by step, carefully assessing the proof attempt from each rubric perspective as detailed as possible. You should output a score for each rubric after your analysis, and provide a brief explanation for each score. The scores should be in the range of 0-10 for each rubric, with 0 indicating no evidence of the criterion and 10 indicating perfect adherence to the criterion. For each rubric, please repeat the proof attempt step by step and analyze it according to the rubric.]
<**ERROR_ANALYSIS**>
<**SCORE**>
[In this field, please provide the sum of the scores from all rubrics, which should be a number between 0 and 80, please use addition to calculate the final score step by step and output the final score in the <**SUM**>`int`<**SUM**> format.
Please first write down the equation for the final score calculation, e.g., "Final Score = int + int + int = int", and then output the final score in the <**SUM**>`int`<**SUM**> format.]
<**SCORE**>
<**FEEDBACK**>
[In this field, please provide the feedback that can help the student improve their proof attempt. The feedback must be based on the error analysis and scores provided above, and should provide clear guidance. Please do not provide the ground truth of the proof directly.]
<**FEEDBACK**>

Figure 25: The prompt with rubrics provided to evaluator to evaluate natural language proof. The detailed <rubrics> is shown in Figure 26.

The Rubrics used in Natural Language Proof Evaluation Prompt

<RUBRICS>
1. Logical Soundness & Step Validity [0-10]
* Content: Whether reasoning is valid, whether logical fallacies or counterexamples exist; whether each reasoning step is correct and reasonable
* Assessment: Check logical derivation relationships, identify fallacy patterns, verify single-step reasoning
2. Completeness of Argument [0-10]
* Content: Whether all necessary cases are covered, whether proof gaps or missing branches exist
* Assessment: Check case coverage, identify unhandled assumptions or boundary conditions
3. Justification & Adequacy [0-10]
* Content: Whether each assertion has sufficient basis, founded on established theorems/axioms/definitions
* Assessment: Verify theoretical support for each key assertion
4. Problem Comprehension & Setup [0-10]
* Content: Whether the problem statement is correctly understood, whether initial assumptions and goals are clear and appropriate
* Assessment: Compare proof setup with problem requirements for consistency
5. Mathematical Rigor [0-10]
* Terminology & Notation: Whether mathematical terms, symbols, definitions, and theorems are used correctly and appropriately
* Computational Accuracy: Whether arithmetic, algebraic, or other mathematical calculation errors are avoided
* Assessment: Verify accuracy of symbolic operations and computational processes
6. Clarity & Presentation [0-10]
* Content: Whether proof structure is logically sound, whether language is clear and comprehensible
* Assessment: Check readability, fluency, and structural organization
7. Relevance & Focus [0-10]
* Content: Whether all statements and calculations are relevant to the proof goal, avoiding redundancy
* Assessment: Identify irrelevant content and reasoning that deviates from the main line
8. Conclusion Completeness [0-10]
* Content: Whether there is a clear concluding statement that summarizes what has been proven
* Assessment: Check correspondence between conclusion and objectives
<RUBRICS>

Figure 26: The rubrics used in prompt shown in Figure 25

Natural Language Proof Evaluation Prompt

<**TASK_TYPE**>
natural_proof_evaluation
<**TASK_TYPE**>

<**PROBLEM**>
{problem_statement}
<**PROBLEM**>

<**PROOF_ATTEMPT**>
{proof_text}
<**PROOF_ATTEMPT**>

Please evaluate this natural language mathematical proof and provide structured feedback.
Your response must follow this exact format:

<**ERROR_ANALYSIS**>
[Provide detailed conceptual analysis of mathematical reasoning, including logical structure assessment, missing components, and error classification]
<**ERROR_ANALYSIS**>

<**SCORE**>
[Provide a numerical score from 0-100]
<**SCORE**>

<**FEEDBACK**>
[Provide educational feedback focusing on mathematical concepts, reasoning gaps, and suggestions for improvement with clear explanations]
<**FEEDBACK**>

Figure 27: The prompt without rubrics provided to evaluator to evaluate natural language proof.

Statement and Proof:

```
import Mathlib
open Real
noncomputable def f (x : R) : R := (1 / cos x) ^2
axiom value_at_pi_over_3 : f (π / 3) = 4
axiom derivative\_of\_f (x : R) : HasDerivAt f (2 * (1 / cos x) ^2 *
     tan x) x
axiom derivative_value_at_pi_over_3 : (2 * (1 / cos (π / 3)) ^2 * tan
     (π / 3)) = 8 * Real.sqrt 3
theorem tangent_line_equation :
∀ x y : R, y = 8 * Real.sqrt 3 * (x - π / 3) + 4 ↔
y - f (π / 3) = (2 * (1 / cos (π / 3)) ^2 * tan (π / 3)) * (x - π /
    3) :=
intro x y
constructor
· intro h
    rw [h, value_at_pi_over_3]
    ring
· intro h
    rw [value_at_pi_over_3] at h
    rw [h]
    ring
```

Feedback:

To fix the proof:
1. In the backward direction, after substituting '$f(\pi)/3) = 4$', the equation becomes '$y - 4 = 8\sqrt{3}\left(x - \frac{\pi}{3}\right)$'. Instead of using 'rw [h]', directly add 4 to both sides using 'rw [← h]' or 'have h' : $y = 8\sqrt{3}\left(x - \frac{\pi}{3}\right) + 4$:= by rw [h]; ring'.
2. Replace the line 'rw [h]' with 'rw [← h]' to correctly isolate 'y'.
3. Ensure the ring tactic is applied to the transformed equation to simplify it to the target form. Example correction for the backward direction:
· intro h
rw [value_at_pi_over_3] at h
have h' : y = 8 * Real.sqrt 3 * (x - π / 3) + 4 := by rw
[h]; ring
rw [h']

Comments:
This feedback is highly beneficial: it correctly identifies the logical error in the proof (the misuse of rw [h] in the backward direction), provides clear and actionable fixes, and includes pedagogical explanations.

Figure 28: A case of *feedback*.

Statement and Proof:

```
theorem aime_1990_p4 (x : ℝ) (h₀ : 0 < x) (h₁ : x ^2 - 10 * x - 29 ≠
    0)
    (h₂ : x ^2 - 10 * x - 45 ≠0) (h₃ : x ^2 - 10 * x - 69 ≠0)
    (h₄ : 1 / (x ^2 - 10 * x - 29) + 1 / (x ^2 - 10 * x - 45) - 2 / (
        x ^ 2 - 10 * x - 69) = 0) :
    x = 13 := by
    have x_pos : 0 < x := h₀
    have h₁ : x ^2 - 10 * x - 29 ≠0 := h₁
    have h₂ : x ^2 - 10 * x - 45 ≠0 := h₂
    have h₃ : x ^2 - 10 * x - 69 ≠0 := h₃
    field_simp [h₁, h₂, h₃] at h₄
    linarith
```

Feedback:

1. **Missing Quadratic Solution**: After determining y = 39, the proof must solve the quadratic equation x^2 - 10x - 39 = 0. Add code to derive x = 13 from this equation.
2. **Use of linarith**: linarith is insufficient here because it only solves linear equations. Instead, use algebraic manipulation (e.g., 'simp', 'rw', or 'ring') to solve the quadratic equation.
3. **Correct Algebraic Steps**: Explicitly show the quadratic solution steps, such as completing the square or using the quadratic formula, to justify x = 13.
4. **Example Fix**: Replace 'linarith' with 'have $h_5 : x^2 - 10 * x - 39 = 0 :=$ by rw $[h_y]$', then solve for x using 'simp' or 'ring'.

Comments:

This feedback is essential because it precisely diagnoses the critical error—the misuse of linarith for solving quadratic equations—while correctly identifying the core mathematical structure (x^2 - 10x - 39 = 0) derived from the hypotheses. It mandates explicit solution of the quadratic equation and provides implementable, pedagogically sound steps to complete the proof. By replacing inadequate tactics with algebraic reasoning and leveraging the positivity constraint to eliminate extraneous roots, the proposed fix delivers a complete and verifiable solution.

Figure 29: A case of *feedback*.

Statement and Proof:

```
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
    have h : 0 < Real.pi := Real.pi_pos
    simp_all [h₁]
```

Feedback:

1. **Add proof for π < 4**: Use a theorem like 'Real.pi_lt_four' (if available) to establish 'Real.pi < 4'. If not, construct the proof using real number properties.
2. **Correctly apply h_1**: Replace 'simp_all [h_1]' with explicit application of h_1 using 'h_1 _ (And.intro (Real.pi_nonneg) (Real.pi_lt_four))' (adjusting theorem names as needed).
3. **Refactor the proof**: Structure the proof as: - Prove '0 $\leq$ Real.pi' from '0 < Real.pi'. - Prove 'Real.pi < 4'. - Use 'h_1' with these conditions to conclude 'f Real.pi = 2'.

Comments: This feedback is highly beneficial as it precisely identifies the critical gap—the missing proof that π < 4 required to apply the function's conditional definition—and resolves it by directing the user to the essential Mathlib theorem Real.pi_lt_four. It corrects the ineffective simp_all tactic by prescribing explicit hypothesis application (h_1 _ ⟨Real.pi_nonneg, Real.pi_lt_four⟩), transforming an incomplete automation attempt into a robust, verifiable proof that adheres strictly to the function's domain conditions while enhancing pedagogical clarity.

Figure 30: A case of *feedback*.

Statement and Proof:

Given positive integers n and k, there is a finite set $A_{(n,k)} = \{\alpha_1, \alpha_2, ..., \alpha_n\}$ where $\alpha_i = (a_{i,1}, a_{i,2}, ..., a_{i,(i-1)k+1})$, $a_{i,1}, a_{i,2}, ..., a_{i,(i-1)k+1} \in \{0,1\}$ and $a_{i,1} + a_{i,2} + ... + a_{i,(i-1)k+1} = 1$ for $i = 1, 2, ..., n$. For an n-term sequence $\Omega : (1, x_1), (2, x_2), ..., (n, x_n)$, if $x_1 = 1$ and $x_{i+1} - x_i \in \{0, 1, ..., k\}$ for $i = 1, 2, ..., n-1$, then it is called a "G-sequence". At this time, denote $G(\Omega) = a_{1,x_1} + a_{2,x_2} + ... + a_{n,x_n}$. Prove that for any set $A_{(n,k)}$, there exists a "G-sequence" Ω such that $(k+1)^{G(\Omega)} > kn$.

Proof:

To prove that for any set $A_{(n,k)}$, there exists a G-sequence Ω such that $(k+1)^{G(\Omega)} > kn$, we can use the probabilistic method and properties of G-sequences.

Key Observations: 1. **G-sequences**: A G-sequence Ω starts at $x_1 = 1$ and each subsequent x_{i+1} satisfies $0 \le x_{i+1} - x_i \le k$. There are $(k+1)^{n-1}$ such sequences since each step has $k+1$ choices.

2. **Function $G(\Omega)$**: $G(\Omega)$ counts the number of positions i where x_i is the position of the 1 in α_i. Let t_i denote the position of the 1 in α_i; then $G(\Omega) = \sum_{i=1}^{n} \mathbf{1}_{\{x_i = t_i\}}$.

3. **Probabilistic Method**: Consider a random G-sequence Ω where each step is chosen uniformly. Define $X = G(\Omega)$. We need to show there exists a sequence with $(k+1)^X > kn$.

Analysis: - The total number of G-sequences is $(k+1)^{n-1}$. - For each sequence, $(k+1)^{G(\Omega)}$ is the product of $(k+1)$ for each i where $x_i = t_i$, and 1 otherwise.

Sum of $(k+1)^{G(\Omega)}$: The sum of $(k+1)^{G(\Omega)}$ over all G-sequences is at least $(k+1)^n$. This is because each sequence contributes at least $k+1$ (since $x_1 = t_1 = 1$ always, so $G(\Omega) \ge 1$), and there are $(k+1)^{n-1}$ sequences.

Pigeonhole Principle: If all sequences had $(k+1)^{G(\Omega)} \le kn$, then the total sum would be at most $kn \times (k+1)^{n-1}$. However, we know the total sum is at least $(k+1)^n$, so:

$$(k+1)^n > kn \times (k+1)^{n-1}$$

Simplifying, this implies $k+1 > kn$, which is true for $n = 1$. For $n \ge 2$, the exponential growth of $(k+1)^n$ compared to the linear growth of kn ensures that there must exist a sequence where $(k+1)^{G(\Omega)} > kn$.

Feedback:

The proof attempt demonstrates an understanding of the probabilistic method and the pigeonhole principle but contains critical errors in the application of these concepts. Here's a breakdown of the feedback:
1. **Conceptual Understanding**: The user correctly identifies the total number of G-sequences and the structure of $G(\Omega)$. They also attempt to use the probabilistic method and the pigeonhole principle, which are valid strategies for such problems.
2. **Logical Gaps**: The core error lies in the incorrect inequality $(k+1)^n > kn \cdot (k+1)^{n-1}$. This inequality is not generally true, and the proof fails to address cases where it does not hold (e.g., $n = 2, k = 1$). The argument also misapplies the pigeonhole principle by assuming that the average value being greater than kn guarantees the existence of a sequence exceeding kn.
3. **Suggestions for Improvement**: - **Re-examine the Inequality**: The proof should address the general case by considering the relationship between $k+1$ and kn for all n and k. For example, when $k+1 \le kn$, alternative methods (e.g., explicit construction of a G-sequence) may be necessary. - **Clarify the Pigeonhole Principle Application**: The argument should explicitly show that the average value of $(k+1)^{G(\Omega)}$ being at least $k+1$ ensures the existence of a sequence with $(k+1)^{G(\Omega)} > kn$, which requires a more rigorous analysis of the total sum and individual contributions. - **Consider Edge Cases**: The proof should test edge cases like $n = 1$ or $k = 1$ to ensure the result holds universally.

Comments: The feedback is highly beneficial as it specifically identifies critical errors in the proof, such as the incorrect application of the inequality $(k+1)^n > kn \cdot (k+1)^{n-1}$ and its failure in cases like $n = 2, k = 1$, while constructively suggesting improvements like re-examining the inequality for all n and k, clarifying the pigeonhole principle usage, and testing edge cases; this precise, actionable guidance helps the author address logical gaps and enhance the proof's rigor, making it both educational and effective for refinement.

Figure 31: A case of *feedback*.