STORM-BORN: A Challenging Mathematical Derivations Dataset Curated via a Human-in-the-Loop Multi-Agent Framework

Wenhao Liu[1*], **Zhenyi Lu**[2*], **Xinyu Hu**[3*], **Jierui Zhang**[1*], **Dailin Li**[4*], **Jiacheng Cen**[5*], **Huilin Cao**[6], **Haiteng Wang**[7], **Yuhan Li**[8], **Kun Xie**[1], **Dandan Li**[1], **Pei Zhang**[1], **Chengbo Zhang**[9], **Yuxiang Ren**[10†], **Xiaohong Huang**[1†], **Yan Ma**[1]

[1]State Key Laboratory of Networking and Switching Technology,
Beijing University of Posts and Telecommunications
[2]Huazhong University of Science and Technology, [3]University of Science and Technology of China,
[4]Dalian University of Technology, [5]Renmin University of China, [6]Shanghai Jiao Tong University,
[7]Beihang University, [8]The Hong Kong University of Science and Technology (Guangzhou),
[9]Peking University, [10]School of Intelligence Science and Technology, Nanjing University
{wenhaoliu,pat,dandl,zhangpei,huangxh,mayan}@bupt.edu.cn

Abstract

High-quality math datasets are crucial for advancing the reasoning abilities of large language models (LLMs). However, existing datasets often suffer from three key issues: outdated and insufficient challenging content, neglecting human-like reasoning, and limited reliability due to single-LLM generation. To address these, we introduce **STORM-BORN**, an ultra-challenging dataset of mathematical derivations sourced from cutting-edge academic papers, which includes dense human-like approximations and heuristic cues. To ensure the reliability and quality, we propose a novel human-in-the-loop, multi-agent data generation framework, integrating reasoning-dense filters, multi-agent collaboration, and human mathematicians' evaluations. We curated a set of 2,000 synthetic samples and deliberately selected the 100 most difficult problems. Even most advanced models like GPT-o1 solved fewer than 5% of them. Fine-tuning on STORM-BORN boosts accuracy by 7.84% (LLaMA3-8B) and 9.12% (Qwen2.5-7B). As AI approaches mathematician-level reasoning, STORM-BORN provides both a high-difficulty benchmark and a human-like reasoning training resource. Our code and dataset are publicly available at `https://github.com/lwhere/STORM-BORN`.

1 Introduction

Mathematical reasoning has emerged as a cornerstone for scaling large language models (LLMs) and probing their upper bounds of intelligence (Shao et al., 2024; Ye et al., 2024; Glazer et al., 2024). Recent advances stem from architectural innovations (McLeish et al., 2024; Li et al., 2025), enhanced pretraining data (Shao et al., 2024; Allal et al., 2025; Wang et al., 2024b), supervised fine-tuning (Yu et al., 2024b; Cobbe et al., 2021; Fan et al., 2025), reinforcement learning (Wang et al., 2024a; Zelikman et al., 2022), and chain-of-thought prompting (Zhang et al., 2022; Lu et al., 2024b; Liu et al., 2024). Current supervised mathematical datasets fall into two categories: numerical reasoning focuses on arithmetic computations yielding numbers (Cobbe et al., 2021; Hendrycks et al., 2021; Glazer et al., 2024), and theorem proving uses formal languages for computer-verifiable proofs (Ying et al., 2024; Wu et al., 2024).

However, existing mathematical datasets face several challenges. *First, they lack nuance and complexity*. Current datasets, like GSM8k (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021), focus on grade-school or competition problems, offering limited complexity. As LLMs excel on these benchmarks, more advanced mathematical reasoning tasks are needed. *Second, human-like reasoning is limited*. While formal languages (de Moura et al., 2015) ensure precise verification in recent datasets (Ying et al., 2024; Wu et al., 2024), they obscure interpretable and intuitive reasoning processes (Chervonyi et al., 2025). *Finally, synthetic data lacks reliable annotations*. While LLMs are used to generate large-scale math data (Yu et al., 2024b; Shao et al., 2024; Chen et al., 2024), they struggle with curating and evaluating expert-level derivations, leading to unreliable step-by-step annotations.

To address these limitations, we focus on a previously underexplored area: *mathematical derivation*. These tasks involve long chains of reasoning (CoT), revisions, and iterative computations, posing significant challenges for current LLMs. More-

*Core contribution.
†Corresponding authors: Xiaohong Huang and Yuxiang Ren.

(I) Numerical Reasoning (PRM-800K)

Question:
An unfair coin lands on heads with probability $3/4$ and tails with probability $1/4$. A heads flip gains \$3, but a tails flip loses \$8. What is the expected worth of a coin flip? Express your answer as a decimal rounded to the nearest hundredth.

Steps:
The expected value is equal to the probability of getting heads times the value of heads plus the probability of getting tails times the value of tails. Let's call the expected value E. So we can write that as $E = 3/4 * 3 + 1/4 * (-8)$. That simplifies to $E = 9/4 - 2$. So the expected value is $E = 1/4 = 0.25$ dollars.

Answer: \$0.25.

(II) Formalized Theory Proving (MiniF2F)

Informal Question:
Expand the following expression: $7(3y + 2)$, show that it is $21y + 14$.

Formal Question:

```
theorem mathd_algebra_182 (y : C)
    : 7 * (3 * y + 2) = 21 * y +
    14 := by
```

Answer:

```
theorem mathd_algebra_182 (y : C)
    : 7 * (3 * y + 2) = 21 * y +
    14 :=
by ring
```

(III) Human-like Derivations (Our STORM-BORN)

Question. Based on Formula (3):

$$\max_{\pi_\theta} \mathbb{E}_{x\sim D, y\sim\pi_\theta(y|x)}[r_\phi(x,y)] - \beta D_{\text{KL}}[\pi_\theta(y|x) \| \pi_{\text{ref}}(y|x)],$$

which enforces a KL-constrained reward maximization, how can we derive Formula (4):

$$\pi_r(y|x) = \frac{1}{Z(x)} \pi_{\text{ref}}(y|x) \exp\Big(\tfrac{1}{\beta} r(x,y)\Big),$$

where $Z(x)$ is the partition function ensuring π_r is a valid probability distribution?

Answer. We optimize the following objective:

$$\max_{\pi} \mathbb{E}_{x\sim\mathcal{D}, y\sim\pi}\left[r(x,y)\right] - \beta\mathbb{D}_{\text{KL}}\left[\pi(y|x) \| \pi_{\text{ref}}(y|x)\right], \tag{1}$$

under any reward function $r(x,y)$, reference model π_{ref}. We now have:

$$\begin{aligned} &\max_{\pi} \mathbb{E}_{x\sim\mathcal{D}, y\sim\pi}\left[r(x,y)\right] - \beta\mathbb{D}_{\text{KL}}\left[\pi(y|x) \| \pi_{\text{ref}}(y|x)\right] \\ &= \max_{\pi} \mathbb{E}_{x\sim\mathcal{D}} \mathbb{E}_{y\sim\pi(y|x)}\left[\log \frac{\pi(y|x)}{\frac{1}{Z(x)}\pi_{\text{ref}}(y|x)\exp\Big(\frac{1}{\beta}r(x,y)\Big)} - \log Z(x)\right], \end{aligned} \tag{2}$$

Observing the result of the final step in the derivation, we notice that π is itself a probability distribution. This naturally raises a question: **what if we could also interpret the denominator as a probability distribution?** If so, it would allow us to reformulate the expression in a way that aligns with the perspective of KL divergence optimization. Based on this idea, we construct a partition function $Z(x)$ as follows:

$$Z(x) = \sum_y \pi_{\text{ref}}(y|x) \exp\left(\frac{1}{\beta} r(x,y)\right) \tag{3}$$

We substitute this $Z(x)$ into the denominator in the previous expression, resulting in:

$$\frac{1}{Z(x)} \pi_{\text{ref}} \exp\left(\frac{1}{\beta} r\right) = \frac{\pi_{\text{ref}}(y|x) \exp\left(\frac{1}{\beta} r(x,y)\right)}{\sum_y \pi_{\text{ref}}(y|x) \exp\left(\frac{1}{\beta} r(x,y)\right)} \tag{4}$$

Note that the partition function is a function of only x and the reference policy π_{ref}, but does not depend on the policy π. We can now define

$$\pi^*(y|x) = \frac{1}{Z(x)} \pi_{\text{ref}}(y|x) \exp\left(\frac{1}{\beta} r(x,y)\right), \tag{5}$$

which is a valid probability distribution as $\pi^*(y|x) \geq 0$ for all y and $\sum_y \pi^*(y|x) = 1$. Since $Z(x)$ is not a function of y, we can then re-organize the final objective in Eq. (2) as:

$$\begin{aligned} &\min_{\pi} \mathbb{E}_{x\sim\mathcal{D}}\left[\mathbb{E}_{y\sim\pi(y|x)}\left[\log \frac{\pi(y|x)}{\pi^*(y|x)}\right] - \log Z(x)\right] \\ &= \min_{\pi} \mathbb{E}_{x\sim\mathcal{D}}\left[\mathbb{D}_{\text{KL}}(\pi(y|x) \| \pi^*(y|x)) - \log Z(x)\right]. \end{aligned} \tag{6}$$

Figure 1: (I) Numerical Reasoning datasets (*e.g.,* PRM-800K) require numerical values, which may be too simplistic for advanced LLMs. (II) Formalized Theory Proving datasets (*e.g.,* MiniF2F) depend on formal languages like Lean, limiting intuitive reasoning and informal generalization. (III) In contrast, our STORM-BORN dataset emphasizes human-like reasoning (highlighted in purple), requiring deep understanding and creativity, more challenging than (I) and more interpretable/generalizable than (II).

over, the data source is both accessible and scalable (*e.g.,* academic papers). Given the complexity of mathematical reasoning, expert annotation is costly and error-prone, while single LLMs often lack reliability. To overcome this, we developed *STORM* (Synergistic Theorem and fORmula Mining), a multi-agent framework that extracts deduction logic, generates question-answer pairs, and performs iterative refinement to ensure reliability.

Using STORM, we synthesized 2,000 high-quality math derivation samples. Recent studies (Ye et al., 2025; Muennighoff et al., 2025) show that dataset quality and complexity are key drivers of reasoning improvements, especially for reasoning-focused models. This is reflected in datasets like Frontier Math (50 samples) (Glazer et al., 2024) and LIMO (817 samples) (Ye et al., 2025), where small and rigorously curated datasets lead to significant reasoning gains. Based on this insight, we further integrate human experts into the curation process, selecting the 100 most challenging problems to form our dataset: **STORM-BORN**, prioritizing reasoning density and correctness. Notably, even advanced models like Deepseek-R1 and GPT-o1-Pro achieve less than 5% accuracy on STORM-BORN, compared to 95% on GSM8K, underscoring its exceptional difficulty. Furthermore, fine-tuning on STORM-BORN leads to strong generalization on numerical reasoning tasks: TinyLlama achieves a 233% relative improvement on MATH, and Qwen improves by 16.7%, despite STORM-BORN not containing explicit numerical reasoning tasks. This demonstrates the broad reasoning skills acquired through our dataset.

Our key contributions can be summarized as follows: (1) We introduce a challenging mathematical derivation dataset curated from recent high-impact papers, featuring complex derivations that demand deep theoretical understanding and creative reasoning. (2) We develop a human-in-the-loop, multi-agent data generation system that extracts complete derivation processes, ensuring human-like reasoning patterns and high-quality, reliable annotations. (3) Extensive human and automatic assessments confirm that even the most advanced LLMs solve fewer than 5% of STORM-BORN problems, highlighting its difficulty. Meanwhile, fine-tuning on our dataset yields strong generalization, particularly for numerical reasoning tasks.

2 Related Work

2.1 Large Language Models for Mathematical Reasoning

Mathematical reasoning has become a critical benchmark for evaluating and improving the capabilities of large language models (LLMs). Advances in this field have been driven by multiple factors, including architectural improvements (McLeish et al., 2024), enhanced pretraining datasets (Shao et al., 2024; Allal et al., 2025; Wang et al., 2024b), supervised fine-tuning (Yu et al., 2024b; Cobbe et al., 2021), reinforcement learning (Wang et al., 2024a; Zelikman et al., 2022), and prompt-based methods such as chain-of-thought reasoning (Ye et al., 2024; Zhang et al., 2022). Frieder et al. (2024) explored LLMs for assisting mathematicians, advocating a hybrid human-model approach. Chang et al. (2023); Fan et al. (2024) evaluated LLMs in mathematical reasoning, noting strengths and limitations. Testolin (2024) and Lu et al. (2023) analyzed deep learning in math problem-solving, highlighting challenges in generalization.

Despite advancements, LLMs in mathematical reasoning remain limited by reliance on dataset-driven learning, leading to brittleness and poor generalization (Ahn et al., 2024; Lu et al., 2024a; Tian et al., 2025). To address this, reinforcement learning has been employed to enhance verification mechanisms (Wang et al., 2024a), while prompt engineering, such as physics-inspired prompting (Ye et al., 2024) and automated chain-of-thought generation (Zhang et al., 2022; Fan et al., 2024), has improved reasoning consistency. These findings highlight the need for structured reasoning techniques alongside architectural and data improvements to further advance mathematical capabilities in LLMs.

2.2 Mathematical Datasets

Mathematical datasets for LLMs can be broadly categorized into numerical reasoning and automated theorem proving (ATP). *For numerical reasoning*, PRM800K (Lightman et al., 2024), GSM8K (Cobbe et al., 2021), and GSM_PLUS (Li et al., 2024a) focus on arithmetic problem-solving, requiring step-by-step derivations. FormulaReasoning (Li et al., 2024b) assesses formula-based numerical reasoning, while GAOKAO (Zhang et al., 2024b) benchmarks LLMs' ability to solve complex mathematical problems in Chinese university entrance exams. *For automated theorem proving*, MiniF2F (Zheng et al., 2022) compiles problems from formal proof assistants, including Metamath (Yu et al., 2024a), Isabelle (Frieder et al., 2024), and Lean (Han et al., 2022). ProofNet (Azerbayev et al., 2023) spans undergraduate-level mathematics, bridging LLMs with formal proof verification. Additionally, DRAW-1K (Upadhyay and Chang, 2017) aids in equation derivation, while Ying et al. (2024); Wu et al. (2024) introduced datasets for Lean, supporting machine-verifiable proof generation.

In contrast, our STORM-BORN dataset focuses on challenging mathematical derivations in natural language, demanding complex reasoning and creativity, and is more likely to contain dense, human-like thinking patterns, such as approximations and heuristic cues.

3 Overall Pipeline

In order to enhance LLMs' reasoning abilities for mathematical expressions found in research papers, we created **STORM-BORN**, a dataset that involves advanced mathematical reasoning. This section describes in detail the construction process of **STORM-BORN**.

3.1 Reasoning-dense Content Filtering

Distinguishing between basic concept explanations and genuinely complex reasoning requires human-like cognitive processes. To ensure our dataset contains more data and of higher quality, a key aspect lies in the selection of data sources—academic

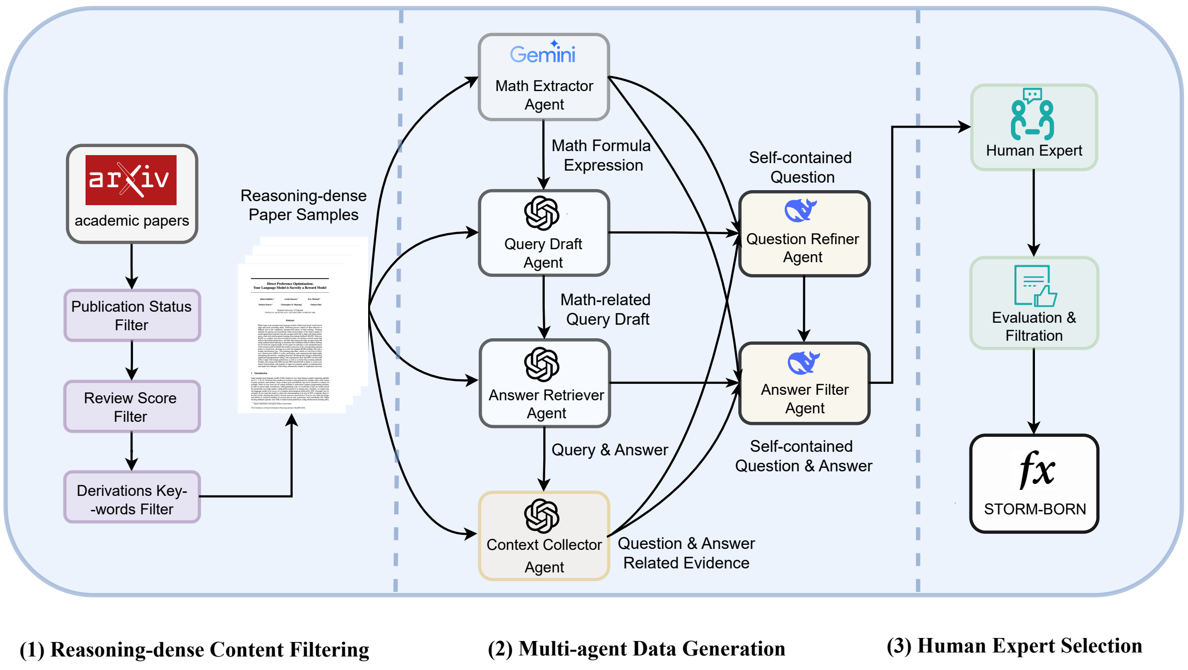

Figure 2: Overview of the data generation framework of STORM-BORN, which consists of three main components: **(1) Reasoning-dense Content Filtering** selects reasoning-dense arXiv papers through linguistic markers and complexity criteria to ensure high-quality mathematical derivations. **(2) Multi-agent Data Generation** orchestrates specialized agents for LaTeX extraction, query formulation, answer retrieval, and context enrichment, culminating in refined mathematical problems. **(3) Human Expert Selection** applies rigorous evaluation criteria to select the most challenging and well-structured problems, resulting in the final STORM-BORN dataset for advancing mathematical reasoning capabilities.

papers. Different papers vary in the amount and quality of data they provide, with some containing extensive mathematical content and detailed proofs and derivation processes, while others do not. Therefore, the focus should be on papers that not only contain a sufficient number of formulas but also provide thorough theorem proofs and derivation processes. More specifically, we select papers based on the following principles.

Publication Status and Review Score. To ensure data reliability, we prioritized papers from reputable journals and conferences, which were peer-reviewed and met stringent acceptance criteria. We also limited the selection to papers published from May 2023 to October 2024 to ensure content freshness and reduce the risk of using outdated material. Additionally, all selected papers had to receive a score higher than "weak accept" from reviewers on the OpenReview platform, ensuring high data quality.

Richness of Mathematical Derivations. We use linguistic markers such as “assume”, “derive”, and “proof” to filter papers that contain detailed derivations and complete sequences of proofs (most of this data comes from the appendices). If the target keywords appear more than five times in a paper, we consider it to have a higher likelihood of being our target paper. This ensures that the filtered papers contain high-quality mathematical reasoning.

3.2 Multi-agent Data Generation

We present a six-agent methodology to generate data. This streamlined workflow (see Fig. 2) ensures that each mathematical expression is accompanied by a coherent proof or derivation, a self-contained question, and a human-like step-by-step answer (a detailed explanation of this process is included below). We spent 200 USD on GPT-o1-Pro and spent about three weeks on prompt engineering. Appendix A contains further details. This multi-agent framework aims to generate high-quality mathematical data by systematically extracting expressions, posing meaningful questions, retrieving and refining answers, gathering requisite background information, and presenting the self-contained results, ultimately providing more transparent insight into mathematical derivations and proofs. In each step, all mathematical symbols and expressions are converted to LaTeX format.

Why not single-agent? We initially experimented with a single-agent approach for data generation, but the results were poor. The task is inherently complex and involves multiple steps. Using a single LLM leads to excessively long prompts with numerous critical points, making it difficult for the model to follow the instructions effectively. By employing a multi-agent system, we can decompose the task into smaller, more manageable components, allowing each LLM agent to focus on a specific step or key point, which improves the results. Additionally, this modular approach provides greater flexibility, making it easier to modify, refine, or integrate new modules for further improvements. In practice, the multi-agent system significantly enhances both the efficiency and quality of data generation.

Math Expression Extractor Agent We utilize lightweight multi-modal LLMs with extensive prompts for accurate LaTeX formula extraction, avoiding the limitations of traditional OCR techniques (He et al., 2024). It uses a multi-modal large language model (MLLM) that can recognize mathematical expressions in text. After collecting these expressions, the original paper and the extracted expressions are forwarded to the Query Draft Agent.

Query Draft Agent We employ the GPT-o1-Pro LLM as our Query Draft Agent, leveraging a well-structured and effective long prompt **exceeding 1k tokens**. It receives the entire paper rather than the chunked paper, which ensures it can comprehensively understand the entire paper. For each expression extracted from the Math Expression Extractor Agent, it generates at least one *query*, focusing on the theorem or formula derivation problems. We also add a few shots to enhance the output format stability. The details of its prompt are in Appendix A.2.

Answer Retriever Agent The Answer Retriever takes the entire paper, a given expression, and its corresponding *query* as input. The Answer Retriever Agent searches the paper for relevant content that can answer the query. Once relevant content is found, it extracts the entire answer directly from the paper rather than make a proof itself to avoid hallucination. Similar to Query Draft Agent, practice has proved that the task of this agent is also difficult and requires a more powerful LLM (*e.g.,* GPT-o1-Pro). The effective prompt we finally use is also relatively long with *nearly 500 tokens*. The details of this prompt are in Appendix A.3.

Context Collector Agent Although Query Draft Agent and Answer Retriever Agent could generate high-quality *query* and *answer*, there still remains the possibility that they lack full information to make them self-contained, which means that the LLMs and humans could answer the question without reading the original paper. The Context Collector captures this information and stores it as *evidence* for the target self-contained *questions* and *answers*.

Question Refiner Agent The goal of this agent is to incorporate the information from the *evidence* into the *query* and *answer*, thereby generating *self-contained question* that can be answered independently without reading the original resource.

Answer Filter Agent Since our goal is to focus on mathematical reasoning, the Answer Filter Agent filters out any irrelevant content after receiving the data processed by the Question Refiner Agent, retaining only the essential information needed to understand how the expression is derived or proven. By filtering out unnecessary data, the subsequent modules can significantly reduce redundant workload and generate self-contained questions and answers.

3.3 Human Expert Selection

Through Multi-agent Data Generation, we obtained 2k samples. We could have directly retained these 2k samples, but our goal was to extract the most challenging and high-quality dataset. To achieve this, we employed a group of expert mathematicians to conduct a rigorous selection process, ultimately arriving at a refined set of 100 samples. We sent the self-contained question and answer generated in (Sec. 3.2) to human experts who are familiar with the reasoning-dense paper samples for selection. Human experts conducted strict audits on data quality, retained data that meets the standards, eliminated data that has no research value, and manually modified and optimized data that was not of borderline quality but could be improved. Each paper was processed by experts for about 30 samples of questions and answers, and the processing of a single paper took about 15 minutes. Through iterative expert feedback and revision, we refined the dataset, ensuring that each sample met the high-quality standards set by our guiding principles. This expert-driven process was critical to ensuring that the dataset reflects complex human-like mathematical reasoning, resulting in the final **STORM-BORN** dataset. This process was guided by the following five core principles: Reasoning Type, Problem Clarity, Derivation Correctness and Reasoning Density.

(Q1) Reasoning Type: *Does the problem demand creative insight and complex reasoning?* Initially, mathematicians determine whether the problem involves genuinely complex reasoning like deriving or proving a formula, as opposed to simple explanation or definition.

(Q2) Problem Clarity: *Is the problem clear, well-defined, and solvable with the existing information?* This step evaluates the explicitness of the problem's goal and conditions. Ambiguities or incomplete queries, where critical context is missing, are flagged for refinement. Human expert intervention is crucial here, as mathematical clarity often requires subjective interpretation, especially when key information is implied or subtly conveyed.

(Q3) Derivation Correctness: *Are all derivation steps logically valid, error-free, and complete?* Mathematicians carefully review each derivation step for correctness, ensuring that all logical transitions are accurate and coherent. This stage presents a significant challenge, as identifying logical errors or omissions often requires a deep theoretical understanding and specialized expertise.

(Q4) Reasoning Density: *Does the reasoning process include sufficient logical steps, exhibit heuristic reasoning cues, and demonstrate trial-and-error similar to human problem-solving?* This requires human expertise to assess whether the reasoning is sufficiently dense, complete, and heuristic. Mathematicians identify patterns in the reasoning that reflect human-like trial-and-error approaches. Missing or incomplete justifications are flagged for further revision.

4 Experiments

4.1 Case Study

In this preliminary case study, we compared three different types of datasets (see Fig. 1): (I) Numerical reasoning datasets such as PRM-800K, which mainly examine numerical calculations, but may be too simple for advanced language models. For example, it can be solved like the expected value of a coin toss, which first calculates the probability of heads and tails, then calculates the payoff. (II) Formal proof datasets, such as Minif2F, which use formal languages such as Lean to describe problems. Although rigorous, they are not easy to understand intuitively and are not easy to associate with real-world scenarios. Moreover, the answer examples can be solved with only one ring. (III) Our proposed STORM-BORN dataset focuses more on human-like reasoning processes and requires deeper understanding, flexible thinking, and complex reasoning. It is not only more challenging than (I), but also more interpretable and general than (II). We selected the DPO (Rafailov et al., 2023) paper as our example, and then the system captured the derivation of important formulas and extracted the complete details of the derivation from the appendix of the paper, which added some human expert thinking, demonstrating the effectiveness of our method in scenarios of complex research.

4.2 Human Evaluation

To validate the challenges of our dataset, we use STORM-BORN as a benchmark and evaluate current state-of-the-art models. Given that our data includes complex mathematical proofs, direct correctness evaluation is difficult. We observe that LLMs often overrate derivation accuracy (*e.g.,* DeepSeek-R1 assigns a perfect score to flawed cases). This makes expert validation essential since mathematical derivations differ fundamentally from numerical reasoning or formal proofs.

In our experiments, we identified consistent error patterns in LLMs, such as algebraic miscalculations, unproven assumptions, symbol misuse, and unjustified logical leaps. Detecting these problems requires reliable expert annotation. To ensure robustness, we conducted multiple evaluation runs, with each model generating three responses per question, independently scored by experts. A fully correct derivation receives 1 point, while partial credit reflects the proportion of key steps completed. We evaluated six leading language models: GPT-o1-Pro, GPT-o1, GPT-o1-Preview, GPT-4o, and DeepSeek-R1. As shown in Fig. 3, the most advanced models (GPT-o1-Pro) achieve just 5% accuracy on STORM-BORN versus 96.4% on MMLU, demonstrating our dataset's unique challenge to mathematical reasoning.

4.3 Automatic Evaluation

To evaluate the fine-tuning effectiveness of our dataset, we assess it in two key aspects: in-domain

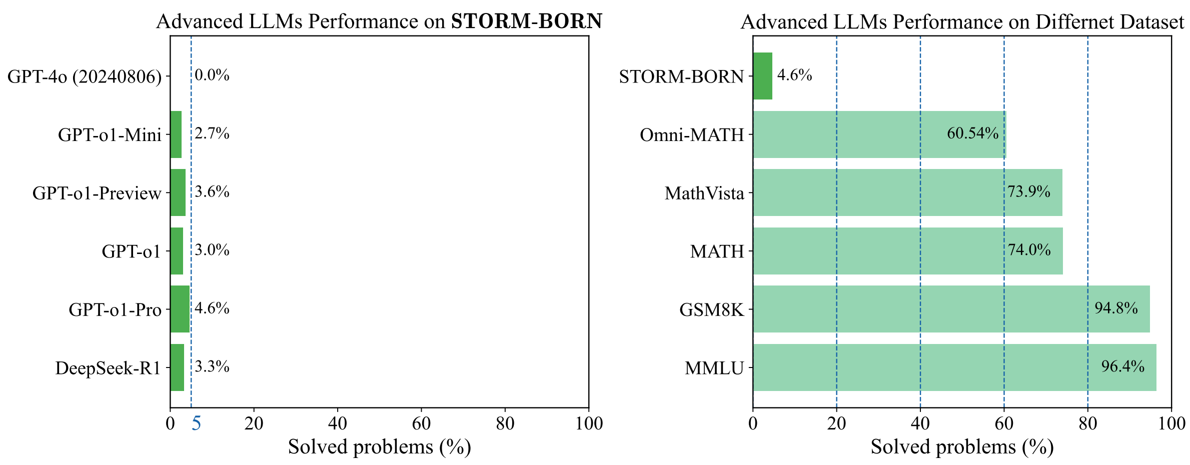

Figure 3: Performance of leading language models on STORM-BORN based on a human expert evaluation. All models show consistently poor performance, with even the best models solving less than 5% of problems. When re-evaluating problems that were solved at least once by any model, GPT-o1-Pro demonstrated the strongest performance across repeated trials.

Model	**GSM8K-0shot**	**GSM8K-8shot**	**MATH-0shot**	**MATH-4shot**
Tiny-LLaMA-1.1B-Chat	1.36%	2.19%	1.20%	2.14%
Tiny-LLaMA-1.1B-Chat (Ours)	**2.05%** (↑ 0.69)	**2.65%** (↑ 0.46)	**4.00%** (↑ 2.80)	**3.46%** (↑ 1.32)
LLaMA2-7B	7.96%	14.33%	1.60%	4.44%
LLaMA2-7B (Ours)	**8.80%** (↑ 0.84)	**16.98%** (↑ 2.05)	**2.60%** (↑ 1.00)	**4.56%** (↑ 0.12)
LLaMA3-8B	15.39%	50.27%	0.12%	17.08%
LLaMA3-8B (Ours)	**42.91%** (↑ 27.52)	45.49% (↓ 4.78)	**7.96%** (↑ 7.84)	13.82% (↓ 3.26)
Qwen2.5-7B	80.67%	83.32%	67.82%	54.42%
Qwen2.5-7B (Ours)	**81.96%** (↑ 1.29)	**83.70%** (↑ 0.38)	**67.96%** (↑ 0.14)	**63.54%** (↑ 9.12)

Table 1: Experimental results of four LLMs on GSM8K and MATH. Models marked with "(Ours)" are additionally fine-tuned on the STORM-BORN dataset. The uparrow (↑) beside each score denotes the absolute accuracy gain obtained after this fine-tuning, relative to the corresponding base model.

Model	**AIME2024**	**AIME2025**
Qwen2.5-7B	0.00%	3.33%
Qwen2.5-7B (Ours)	**3.33%** (↑ 3.33)	**6.67%** (↑ 3.34)
Qwen2.5-32B	20.00%	15.00%
Qwen2.5-32B (Ours)	**23.33%** (↑ 3.33)	15.00% (↑ 0.00)

Table 2: Accuracy (%) of Qwen 2.5 7B and 32B on the AIME 2024 and AIME 2025. Models marked with "(Ours)" are additionally fine-tuned on the STORM-BORN dataset. The uparrow (↑) beside each score denotes the absolute accuracy gain obtained after this fine-tuning, relative to the corresponding base model.

Method	**Correctness**	**Completeness**	**Similarity**	**Avg.**
Qwen2.5-7B	1.10	1.14	0.50	0.91
Qwen2.5-7B (Ours)	**1.23**	**1.32**	**0.66**	**1.07**

Table 3: Formula Derivation Performance

formula derivation performance and out-of-domain numerical reasoning generalization ability.

Numerical Reasoning To assess the generalization of STORM-BORN on mathematical reasoning, we conduct full fine-tuning experiments on four most popular models: TinyLLaMA-1.1B-chat (Zhang et al., 2024a), LLaMA2-7B (Touvron et al., 2023), LLaMA3-8B (Grattafiori et al., 2024) and Qwen2.5-7B (Qwen et al., 2025), evaluating their performance on GSM8K (Cobbe et al., 2021), MATH (Hendrycks et al., 2021) and AIME. For few-shot evaluation, we follow the setup from Touvron et al. (2023), using 8-shot for GSM8K and 4-shot for MATH. Due to the extreme difficulty of AIME and the negligible performance of LLaMA-based models (scoring near zero), we evaluate only Qwen2.5-7B on this benchmark.

As shown in Table 1, fine-tuning solely on 100 STORM-BORN samples significantly boosts performance across benchmarks:relative improvement of 233.3% MATH for Tiny-LLaMA-1.1B-chat, 62.5% for LLaMA2-7B, absolute improvement of 7.84% MATH for LLaMA3-8B, and 9.12% for Qwen2.5-7B. These results demonstrate that

Method	GSM8k-0shot	GSM8k-8shot	Math-0shot	Math-4shot
LLaMA2-7B	7.96	14.33	1.60	4.44
LLaMA2-7B + top-100	**8.80**	**16.98**	**2.60**	**4.56**
LLaMA2-7B + top-500	8.11	15.09	2.40	4.52
LLaMA2-7B + 2k	6.70	14.87	2.28	4.46
LLaMA2-7B + fullset	5.16	14.10	2.58	4.92

Table 4: Quality Ablation Performance

STORM-BORN generalizes well to out-of-domain numerical reasoning tasks. Moreover, the improvement increases with model capability, indicating that stronger foundation models benefit more from our dataset. Table 2 illustrates that even on the highly challenging AIME benchmark, our dataset continues to drive improvements in model performance. This highlights the strong generalization and reasoning capabilities gained through fine-tuning on STORM-BORN, particularly for advanced problem-solving skills required in expert-level mathematics.

Formula Derivation We evaluated our method on a formula derivation test set extracted from NuminaMath-1.5 (LI et al., 2024). We fully fine-tune Qwen2.5-7B (Qwen et al., 2025) on STORM-BORN and use Deepseek-R1 for evaluation. We assessed correctness, completeness, and similarity to ground-truth proofs (each scored on a 0–2 scale) by comparing both ground-truth proofs and predicted derivations. The evaluation prompt is detailed in Appendix C. The results show a relative improvement of 17.58%, with the average score increasing from 0.91 to 1.07. This demonstrates that our dataset directly benefits formula derivation reasoning.

4.4 Quality Ablation

Our dataset prioritizes quality and difficulty, which led us to reduce the initial 2,000 automatically generated samples down to a rigorously curated set of 100 high-quality examples. To evaluate how quality and quantity affect fine-tuning performance, we slightly relaxed our selection criteria (reasoning density), to construct larger subsets: top-100, top-500, and a full 2k automated set. We then conducted a comparative experiment across these subsets.

As shown in Table 4, we can observe: (1) LLaMA2-7B + top-100 outperforms larger datasets: On GSM8K, it achieves higher accuracy in both zero-shot (8.80 vs. 8.11) and 8-shot (16.98 vs. 15.09) settings compared to the top-500 and 2k sets, demonstrating that small, high-quality data enhances reasoning more effectively than larger, lower-quality data. (2) Full unfiltered set harms performance: LLaMA2-7B trained on the full, uncurated set underperforms the base model (*e.g.,* GSM8K zero-shot: 5.16 vs. 7.96). In contrast, the 2k set generated by our pipeline without human validation still improves over the base model (*e.g.,* 14.87 vs. 14.33), confirming the effectiveness of our synthesis framework.

5 Conclusion

In conclusion, we present **STORM-BORN**, a novel dataset designed to address the limitations of existing mathematical derivation datasets. Curated from recent top-tier academic papers via the arXiv repository, STORM-BORN is both nuanced and scalable, while avoiding data contamination. Unlike isolated steps, we capture full derivations to preserve logical flow and encourage deep theoretical reasoning. Using a human-in-loop and multi-agent LLM framework *STORM*, we generate problems requiring at least three reasoning steps, ensuring complexity and creativity. Expert evaluations ensure reliable annotations. Empirical results highlight the dataset's challenge, with advanced LLMs like GPT-o1-Pro solving fewer than 5% of the problems, compared to 95% accuracy on GSM8K. Additionally, STORM-BORN demonstrates strong generalization capabilities, offering a high-difficulty evaluation benchmark for AI's approach to mathematician-level reasoning. In this way, STORM-BORN will be a pioneer in deriving expert evaluation and provide a human-evaluation and auto-evaluation template for future work.

Limitations

This study addresses an important gap in the field, but it also faces certain limitations. Specifically, the automated evaluation of data quality remains challenging, as our focus on complex mathematical derivations rather than numerical computing makes quality assessment difficult (a problem also noted by Glazer et al. (2024)). Currently, we rely primarily on a carefully designed multi-agent curation pipeline and manual inspection by mathematicians. However, with the rapid advancement and scaling of LLMs, we believe that in the future, LLMs can be fully employed to automate this process, iteratively improving and optimizing it.

Ethics Statement

The dataset construction process in this study strictly adheres to ethical guidelines and fully complies with relevant legal regulations. We obtain publicly accessible, high-quality academic papers and utilize a combination of multimodal models and human evaluation feedback for data processing and optimization, ensuring data quality and reliability before generating the final dataset. The entire data collection and processing workflow is transparent and traceable, with all papers sourced from legal and publicly available channels, guaranteeing compliance and traceability of data. The dataset constructed in this study is intended solely for academic research and experimental purposes, with no involvement in commercial applications or risk of sensitive information leakage.

References

Janice Ahn, Rishu Verma, Renze Lou, Di Liu, Rui Zhang, and Wenpeng Yin. 2024. Large language models for mathematical reasoning: Progresses and challenges. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: Student Research Workshop*, pages 225–237, St. Julian's, Malta. Association for Computational Linguistics.

Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Hynek Kydlíček, Agustín Piqueres Lajarín, Vaibhav Srivastav, Joshua Lochner, Caleb Fahlgren, Xuan-Son Nguyen, Clémentine Fourrier, Ben Burtenshaw, Hugo Larcher, Haojun Zhao, Cyril Zakka, Mathieu Morlon, Colin Raffel, Leandro von Werra, and Thomas Wolf. 2025. Smollm2: When smol goes big – data-centric training of a small language model. *Preprint*, arXiv:2502.02737.

Zhangir Azerbayev, Bartosz Piotrowski, Hailey Schoelkopf, Edward W. Ayers, Dragomir Radev, and Jeremy Avigad. 2023. Proofnet: Autoformalizing and formally proving undergraduate-level mathematics.

Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, Wei Ye, Yue Zhang, Yi Chang, Philip S. Yu, Qiang Yang, and Xing Xie. 2023. A survey on evaluation of large language models. *Preprint*, arXiv:2307.03109.

Nuo Chen, Yuhan Li, Jianheng Tang, and Jia Li. 2024. Graphwiz: An instruction-following language model for graph computational problems. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 353–364.

Yuri Chervonyi, Trieu H. Trinh, Miroslav Olšák, Xiaomeng Yang, Hoang Nguyen, Marcelo Menegali, Junehyuk Jung, Vikas Verma, Quoc V. Le, and Thang Luong. 2025. Gold-medalist performance in solving olympiad geometry with alphageometry2. *Preprint*, arXiv: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. 2021. Training verifiers to solve math word problems. *Preprint*, arXiv:2110.14168.

Leonardo Mendonça de Moura, Soonho Kong, Jeremy Avigad, Floris van Doorn, and Jakob von Raumer. 2015. The lean theorem prover (system description). In *CADE*.

Chenghao Fan, Zhenyi Lu, Sichen Liu, Chengfeng Gu, Xiaoye Qu, Wei Wei, and Yu Cheng. 2025. Make lora great again: Boosting lora with adaptive singular values and mixture-of-experts optimization alignment. *Preprint*, arXiv:2502.16894.

Chenghao Fan, Zhenyi Lu, Wei Wei, Jie Tian, Xiaoye Qu, Dangyang Chen, and Yu Cheng. 2024. On giant's shoulders: Effortless weak to strong by dynamic logits fusion. *Preprint*, arXiv:2406.15480.

Simon Frieder, Julius Berner, Philipp Petersen, and Thomas Lukasiewicz. 2024. Large language models for mathematicians. *Preprint*, arXiv:2312.04556.

Elliot Glazer, Ege Erdil, Tamay Besiroglu, Diego Chicharro, Evan Chen, Alex Gunning, Caroline Falkman Olsson, Jean-Stanislas Denain, Anson Ho, Emily de Oliveira Santos, Olli Järviniemi, Matthew Barnett, Robert Sandler, Matej Vrzala, Jaime Sevilla, Qiuyu Ren, Elizabeth Pratt, Lionel Levine, Grant Barkley, Natalie Stewart, Bogdan Grechuk, Tetiana Grechuk, Shreepranav Varma Enugandla, and Mark Wildon. 2024. Frontiermath: A benchmark for evaluating advanced mathematical reasoning in ai. *Preprint*, arXiv:2411.04872.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur

Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha

White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Jesse Michael Han, Jason Rute, Yuhuai Wu, Edward Ayers, and Stanislas Polu. 2022. Proof artifact cotraining for theorem proving with language models. In *International Conference on Learning Representations*.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. 2024. OlympiadBench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3828–3850, Bangkok, Thailand. Association for Computational Linguistics.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *Preprint*, arXiv:2103.03874.

Jia LI, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Costa Huang, Kashif Rasul, Longhui Yu, Albert Jiang, Ziju Shen, Zihan Qin, Bin Dong, Li Zhou, Yann Fleureau, Guillaume Lample, and Stanislas Polu. 2024. Numinamath. `[https://huggingface.co/AI-MO/NuminaMath-1.5](https://github.com/project-numina/aimo-progress-prize/blob/main/report/numina_dataset.pdf)`.

Qintong Li, Leyang Cui, Xueliang Zhao, Lingpeng Kong, and Wei Bi. 2024a. GSM-plus: A comprehensive benchmark for evaluating the robustness of LLMs as mathematical problem solvers. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2961–2984, Bangkok, Thailand. Association for Computational Linguistics.

Xiao Li, Bolin Zhu, Sichen Liu, Yin Zhu, Yiwei Liu, and Gong Cheng. 2024b. Formulareasoning: A dataset for formula-based numerical reasoning. *Preprint*, arXiv:2402.12692.

Zongzhao Li, Jiacheng Cen, Bing Su, Wenbing Huang, Tingyang Xu, Yu Rong, and Deli Zhao. 2025. Large language-geometry model: When llm meets equivariance. *arXiv preprint arXiv:2502.11149*.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*.

Wenhao Liu, Tianxing Bu, Erchen Yu, Dailin Li, Ding Ai, Zhenyi Lu, and Haoran Luo. 2024. Optimizing few-shot learning: From static to adaptive in qwen2-7b. In *Submitted to Amazon KDD Cup 2024 Workshop*. Under review.

Pan Lu, Liang Qiu, Wenhao Yu, Sean Welleck, and Kai-Wei Chang. 2023. A survey of deep learning for mathematical reasoning. *Preprint*, arXiv:2212.10535.

Zhenyi Lu, Chenghao Fan, Wei Wei, Xiaoye Qu, Dangyang Chen, and Yu Cheng. 2024a. Twin-merging: Dynamic integration of modular expertise in model merging. *Advances in Neural Information Processing Systems*, 37:78905–78935.

Zhenyi Lu, Jie Tian, Wei Wei, Xiaoye Qu, Yu Cheng, Wenfeng xie, and Dangyang Chen. 2024b. Mitigating boundary ambiguity and inherent bias for text classification in the era of large language models. *Preprint*, arXiv:2406.07001.

Sean McLeish, Arpit Bansal, Alex Stein, Neel Jain, John Kirchenbauer, Brian R. Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, Jonas Geiping, Avi Schwarzschild, and Tom Goldstein. 2024. Transformers can do arithmetic with the right embeddings. *Preprint*, arXiv:2405.17399.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. 2025. s1: Simple test-time scaling. *Preprint*, arXiv:2501.19393.

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. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *Preprint*, arXiv:2402.03300.

Alberto Testolin. 2024. Can neural networks do arithmetic? a survey on the elementary numerical skills of state-of-the-art deep learning models. *Applied Sciences*, 14(2):744.

Jie Tian, Xiaoye Qu, Zhenyi Lu, Wei Wei, Sichen Liu, and Yu Cheng. 2025. Extrapolating and decoupling image-to-video generation models: Motion modeling is easier than you think. *Preprint*, arXiv:2503.00948.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models. *Preprint*, arXiv:2307.09288.

Shyam Upadhyay and Ming-Wei Chang. 2017. Annotating derivations: A new evaluation strategy and dataset for algebra word problems. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 494–504, Valencia, Spain. Association for Computational Linguistics.

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

Zengzhi Wang, Xuefeng Li, Rui Xia, and Pengfei Liu. 2024b. Mathpile: A billion-token-scale pretraining corpus for math. *Preprint*, arXiv:2312.17120.

Zijian Wu, Jiayu Wang, Dahua Lin, and Kai Chen. 2024. Lean-github: Compiling github lean repositories for a versatile lean prover. *Preprint*, arXiv:2407.17227.

Tian Ye, Zicheng Xu, Yuanzhi Li, and Zeyuan Allen-Zhu. 2024. Physics of language models: Part 2.1, grade-school math and the hidden reasoning process. *Preprint*, arXiv:2407.20311.

Yixin Ye, Zhen Huang, Yang Xiao, Ethan Chern, Shijie Xia, and Pengfei Liu. 2025. Limo: Less is more for reasoning. *Preprint*, arXiv:2502.03387.

Huaiyuan Ying, Zijian Wu, Yihan Geng, Jiayu Wang, Dahua Lin, and Kai Chen. 2024. Lean workbook: A large-scale lean problem set formalized from natural language math problems. *Preprint*, arXiv:2406.03847.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng YU, Zhengying Liu, Yu Zhang, James Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2024a. Metamath: Bootstrap your own mathematical questions for large language models. In *The Twelfth International Conference on Learning Representations*.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T. Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2024b. Metamath: Bootstrap your own mathematical questions for large language models. *Preprint*, arXiv:2309.12284.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. 2022. Star: Bootstrapping reasoning with reasoning. *Preprint*, arXiv:2203.14465.

Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. 2024a. Tinyllama: An open-source small language model. *Preprint*, arXiv:2401.02385.

Xiaotian Zhang, Chunyang Li, Yi Zong, Zhengyu Ying, Liang He, and Xipeng Qiu. 2024b. Evaluating the performance of large language models on gaokao benchmark. *Preprint*, arXiv:2305.12474.

Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. 2022. Automatic chain of thought prompting in large language models. *Preprint*, arXiv:2210.03493.

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

A Workload and Prompts

We invested a lot of work, energy, and time in this research. Our goal is to generate high-quality formula derivation and question-answering. At first glance, this seems to be a simple task, but in fact it involves extremely complex and extensive workload. Initially, we explored various technical solutions, such as optical character recognition (OCR), but when using OCR for formula recognition and extraction, we often encountered incomplete positioning (only part of the formula was framed out), resulting in inaccurate formula extraction. After repeated comparisons and experiments, we finally chose the method of multi-agent large language model (LLM) collaboration, which has consumed some time and energy.

The biggest challenge appeared in the prompt design and optimization stage. Practice has shown that LLM will encounter a series of problems, such as identifying key data in long texts, following instructions, and producing stable output. To solve these difficulties, we continuously refined the overall workflow and assigned complex tasks to multiple appropriate numbers of agents (see Fig. 2) for collaborative execution. At the same time, the prompts of each agent were modified, iterated, and verified for multiple rounds. This process is tedious and time-consuming, and consumes a lot of energy.

Regarding manual evaluation and feedback, each paper required individuals with relevant academic background to read, assess, and provide feedback on the generated data, which increases labor and time costs.

For resource costs and time costs, please see Appendix B.

Thanks to this painstaking and systematic workflow, we were finally able to obtain high-quality question-answering data. We will introduce our prompts below, hoping to provide further insight into the complexity of this study, the extensive workload involved, and our efforts to overcome a variety of challenges.

A.1 Math Expression Extractor Agent

We encountered many problems in the process, such as: the set of extracted mathematical expressions omitted important items, contained unnecessary items and repeated items; the output latex format did not meet the requirements. To solve these problems, we added new rules to the prompt and repeatedly verified the effect in practice, **and iterated continuously**. Through repeated iterations in practice, these problems were solved, which enables the MLLM to follow the instructions to extract all important mathematical expressions (formulas, theorems, lemmas, etc.), ignore unimportant mathematical expressions (such as intermediate expressions that appear in the derivation process, mathematical content inserted in the paragraph), and ensure that the output expression is in the correct format.

Prompt of Math Expression Extractor

```
"""Read the paper, then:

1. Formula Recognition:
- Identify all mathematical formulas, theorems, lemmas, and corollaries in
    the paper. Especially Numbered formulas.Retain the formula's number (if
    any).
- For formulas without explicit labels (i.e., those not labeled as "theorem,
    " "lemma, " or "corollary"), classify them as "formula."
- Required types of formulas to recognize:
    - Numbered formulas.
    - Formulas that appear on separate lines (for example, occupying a line
    or multiple lines by themselves in the paper).
- Ignore:
    - Formulas that appear in the middle of a paragraph without separate
    lines or numbers.
- Make sure there are no duplicates in the results (duplicates refer to
    formulas that are exactly the same after conversion to LaTeX. If the same
    formula appears in the paper under different numbers, treat them as the
    same formula).
```

```
2. LaTeX Conversion (Convert the formulas identified in step 1 into LaTeX
    format strings):
- Symbols: Convert mathematical symbols accurately.
- Subscripts and superscripts: Convert subscripts and superscripts correctly.
- Uppercase and lowercase: Preserve the original variable and constant casing.
- Formula structure: Keep the entire structure of the formula intact.
- Formula numbering: Retain the formula's number (if any).
- Italics: For italicized variables in the text, wrap them with \textit{} in
    LaTeX.
- Math environment: Use `$ ... $` for inline formulas and `$$ ... $$` for block
    (display) formulas.
- Additional conditions: Check whether the paper includes definitions or
    explanations immediately following the formula (for example, "where X is
    ...") and incorporate them if present.

3. JSONL Output:
- Output all converted LaTeX strings in multi-line JSONL format so they can
    be parsed line by line.
- Each line should be a JSON object whose key is the type of the formula
    ("formula", "lemma", "theorem", "corollary", etc.) and whose value is the
    LaTeX string obtained from step 2.
- Be sure to follow the requirements in step 2!

Ensure the formulas are exactly the same as in the original text!"""
```

A.2 Query Draft Agent

The more difficult task also leads to more problems encountered in the process, such as the generated questions are too rigid, the questions lack prerequisites, and only the formula reference number is output without the original formula which emphasizes the need of Context Collector Agent and Question Refiner Agent.

Prompt of Query Draft

```
"""I will provide you with a dataset extracted from this paper, in JSONL
    format. Each entry is a dictionary whose keys are "formula, " "lemma, "
    "theorem, " etc., representing the category of the mathematical
    expression, and whose values contain a mathematical expression in LaTeX
    format, extracted from the paper.

Carefully read and understand the paper's content, especially the parts
    related to each formula in the JSONL. For each formula, please complete
    the following steps:

---

Step 1:
Locate where the formula is first defined or fully derived in the paper, and
    use the relevant context to extract all the direct necessary conditions
    for deriving or proving that formula. These preconditions include, but
    are not limited to:

1. Which other formulas this formula is derived from or depends on. For each
    such formula, record its full content (in LaTeX format), its numbering
    (if any), and its name (if any).
2. Relevant problem settings.
3. The specific meaning of symbols or variables involved in the formula.

---

Step 2:
Based on the extracted preconditions, generate a complete question that
    clearly asks how to derive or prove the formula. The question should
    include:
```

```
1. The formula itself: Present the full content of this formula (in LaTeX
    format). Do not only reference its number.
2. The preconditions: Explicitly integrate the preconditions extracted from
    the paper into the question. List out the full contents of all the
    formulas it depends on and reference them by their respective numbers or
    names. Do not produce a question such as "What are the preconditions?"

The form of the question must meet the following requirements:

- If a formula is derived from one or more other formulas, you must
    explicitly list the full content (in LaTeX) of these preceding formulas
    and reference them by their numbers or names, and explain how the current
    formula is derived from them. For example, if the paper contains Formula
    3 (content: X) and Formula 4 (content: Y), and Formula 4 is derived from
    Formula 3, then the generated question should be:

"Based on Formula 3: X, how can we derive Formula 4: Y?"

- If the formula is a theorem, lemma, or corollary, please generate a
    question asking how to prove it, for example:

"How can we prove Lemma 1: X is true?"

Note: The question must be structured and logical, clearly showing the
    derivation or proof process of the formula and explicitly reflecting the
    dependency between formulas while fully presenting all related formulas.

---

Step 3:
Match each formula with its corresponding question and output the result in
    multi-line JSONL format.

Each data entry should be a dictionary containing the following two key-value
    pairs:

1. Formula type:
- The key is "formula, " "lemma, " "theorem, " etc.
- The value is the LaTeX content of the formula.
2. Generated question:
- The key is "query."
- The value is the complete question generated according to Step 1 and Step 2.

---

Important Notes:
1. Format Requirements:
- Ensure the output is in JSONL format, with each line corresponding to one
    data entry.
2. Formula Accuracy:
- If the question contains mathematical expressions, convert them into LaTeX
    format. Make sure they align with the original mathematical meaning.
    Minor formatting differences can be ignored.
3. LaTeX Conversion (Converts the mathematical expressions contained in the
    problem to strings in LaTeX format):
- Symbols: Convert mathematical symbols accurately.
- Subscripts and superscripts: Convert subscripts and superscripts correctly.
- Uppercase and lowercase: Preserve the original variable and constant casing.
- Formula structure: Keep the entire structure of the formula intact.
- Formula numbering: Retain the formula's number (if any).
- Italics: For italicized variables in the text, wrap them with \textit{} in
    LaTeX.
- Math environment: Use `$ ... $` for inline formulas and `$$ ... $$` for block
    (display) formulas.
4. Completeness of Preconditions:
- The question content must include all direct necessary conditions.
    Particularly, indicate which other formulas the current formula is
    derived from or depends on, and clearly specify the entire content,
    numbering, or name of those referenced formulas. Do not produce questions
    such as "What are the preconditions?"

---

Examples:
Here are some example questions and their corresponding output formats for
    reference:

- Suppose the paper contains the following formula:
{"lemma": "Lemma 1.  The function $f (x)$ is continuous."}
The generated question might be:
{"query":"How can we prove Lemma 1: The function $f (x)$ is continuous. is
    true?"}

- Suppose the paper contains the following formula:
{"formula": "y = mx + b"}
and it is explained that this formula is derived from y = f (x) and f (x) =
    mx + b. Then the generated question might be:
{"query":"Based on the formulas: $y = f (x)$ and $f (x) = mx + b$, how can we
    derive the formula: $y = mx + b$?"}

- Suppose the paper contains the following formula:
{"formula": "$$\\pi_r (y | x) = \\frac{1}{Z (x)} \\pi_{ref}(y | x) \\exp
    (\\frac{1}{\\beta} r (x, y))$$"}
and it is explained that this formula is derived from Formula 3, $KL (\\pi_r
    (y|x) || \\pi_{ref}(y|x)) \\leq \\epsilon$. Then the generated question
    should be:
{"query":"Based on Formula 3: $KL (\\pi_r (y|x) || \\pi_{ref}(y|x)) \\leq
    \\epsilon$, how can we derive Formula: $\\pi_r (y | x) = \\frac{1}{Z (x)}
    \\pi_{ref}(y | x) \\exp (\\frac{1}{\\beta} r (x, y))$?"}

The dataset is as follows:\n
```

A.3 Answer Retriever Agent

In order to solve the problems encountered in the process, such as: **the answer is not extracted from the original text but the large model generates the answer itself**, the answer retrieved in this agent may lack the important complete proof process in the appendix, or is a summary of the answer in the original text, the effective prompt we finally get is also relatively long with nearly 500 tokens.

Prompt of Answer Retriever

```
"""I will provide a JSONL-format dataset extracted from this paper. Each
    piece of data in the dataset is a dictionary containing two main
    key-value pairs:
1. **Formula-related keys ("formula", "lemma", "theorem", etc.)** indicating
    the type of mathematical expression; the value is the LaTeX-formatted
    mathematical expression extracted from the paper.
2. **query**, whose value is a question generated by a large model based on
    the paper and the mathematical expression.

Please process this dataset according to the following steps and requirements.

---

### Step One:
For the "expression" and "query" in each piece of data, determine whether the
    answer to that question can be found in the paper. The specific steps are
    as follows:

1. **Find the first occurrence**
   - Locate where the expression first appears in the paper and check the
    surrounding context for relevant clues.
   - If there are any references or citations, follow those as well.
```

```
2. **Check the appendix and other sections**
   - Search the paper's appendix or other relevant chapters to see if the
    proof or derivation steps for that expression are provided. This may well
    be the answer to the question.

3. **Confirm feasibility**
   - If the paper does not include any relevant content addressing the
    question, you may skip this expression and proceed to the next one.
   - If the paper does indeed contain content that can answer the question,
    extract the relevant content from the original text.

When extracting the answer, please note the following requirements:
- **Completeness**: The extracted answers should cover all the relevant steps
    needed to solve the problem in the paper.
- **Consistency**: Include only content from the original text in the answer
    (you may make minimal necessary edits for coherence, but do not change
    the original meaning). Avoid adding extra content or descriptions not
    found in the original text.
- **Citation handling**: If the answer cites other formulas or theorems from
    the paper, also include their original content in the derivation or proof
    process, rather than leaving only references or labels.
- **LaTeX conversion**: Ensure all mathematical expressions are converted to
    the same LaTeX format as in the original text, including:
  - Accuracy of symbols, subscripts, superscripts, and capitalization.
  - Preserving the original structure and numbering (if any).
  - Using \textit{} for italicized variables.
  - Using $...$ for inline math expressions and $$...$$ for display math
    expressions.

---

### Step Two:
Match the answers extracted in Step One with the corresponding entries in the
    dataset, and add a new key-value pair to form a new data record. The
    specific requirements are:

- For each original data entry, add a new key called `whole_label`, whose
    value is the LaTeX-formatted answer content extracted from the paper.
- Output format must be **multi-line JSONL**, one piece of data per line:
  1. The original two key-value pairs remain unchanged and must not be
    modified.
  2. Add the `whole_label` key as the third key-value pair.

---

### Output Requirements:
1. **Multi-line JSONL format**: One data entry per line.
2. **Accuracy of content**: Formulas must match the original text of the
    paper exactly, with correct symbols, subscripts, superscripts, and
    capitalization.
3. ** Content consistency ** : Only retain the original content in the answer
    (you can make a small amount of necessary cohesive editing, but do not
    change the original meaning), and try to avoid adding additional content
    or descriptions that do not appear in the original.
---

### Note:
- Please strictly follow the above requirements to avoid omitting any key
    content.
- Ensure there are no errors or incomplete parts in the output text.

---

Below is the dataset:
"""
```

B Resource and Time Costs

At the outset, it is important to highlight the considerable workload entailed in our approach, with the associated resource and time costs reflecting the extensive efforts required for its implementation.

B.1 Resource Costs

To support the multi-agent system, we subscribed to GPT-o1-Pro for a one-month period at an approximate cost of 200 USD. For simpler tasks, such as Math Expression Extraction and Answer Filtering, we utilize free LLMs as agents. For more complex tasks, such as Query Generation and Answer Retrieval, we rely on the paid GPT-o1-Pro model to ensure enhanced performance and accuracy. An illustrative example is provided below. After the Math Expression Extraction step, using GPT-o1-Pro for both Query Generation and Answer Retrieval yielded the results shown in Appendix B.1.1. In contrast, using the gemini-2.0-flash-exp model for the same tasks resulted in Appendix B.1.2. Furthermore, when using the query generated by GPT-o1-Pro and applying gemini-2.0-flash-exp for Answer Retrieval, the result was Appendix B.1.3. This comparison clearly illustrates the necessity of using more powerful models like GPT-o1-Pro for complex tasks.

B.1.1 GPT-o1-Pro Results

Here we provide the results of GPT-o1-Pro as follows:

GPT-o1-Pro as Query Draft and Answer Retriever.

formula:

```
"$$\\pi_{r}(y | x)=\\frac{1}{Z (x)} \\pi_{r e f}(y | x) \\exp
    (\\frac{1}{\\beta} r (x, y)), $$(4)"
```

query:

```
"Based on Formula (3): $$\\max _{\\pi_{\\theta}} \\mathbb{E}_{x \\sim D, y
    \\sim \\pi_{\\theta}(y|x)}[r_{\\phi}(x, y)]-\\beta D_{K
    L}[\\pi_{\\theta}(y | x) || \\pi_{r e f}(y | x)], $$ which enforces a
    KL-constrained reward maximization, how can we derive Formula (4): $$
    \\pi_{r}(y | x)=\\frac{1}{Z (x)} \\pi_{r e f}(y | x) \\exp\\Bigl
    (\\tfrac{1}{\\beta} r (x, y)\\Bigr), $$ where $Z (x)$ is the partition
    function ensuring $\\pi_{r}$ is a valid probability distribution?"
```

answer:

```
"In the paper's Appendix A.1 (pp. 15-16), ``Deriving the Optimum of the
    KL-Constrained Reward Maximization Objective, '' the authors
    write:\n\n``[We] optimize the following objective:\n$$\\max_{\\pi}\\,
    \\mathbb{E}_{x\\sim D, y\\sim\\pi (y\\mid x)}\\bigl[r (x, y)\\bigr]\\;
    -\\; \\beta\\, D_{\\mathrm{KL}}\\bigl[\\pi (y\\mid x)\\, \\|\\,
    \\pi_{\\mathrm{ref}}(y\\mid x)\\bigr].$$\n\n... We now
    have:\n$$\\max_{\\pi}\\, \\mathbb{E}_{x\\sim D, y\\sim \\pi (y\\mid
    x)}\\bigl[r (x, y)\\; -\\; \\log\\frac{\\pi (y\\mid
    x)}{\\pi_{\\mathrm{ref}}(y\\mid x)}\\bigr]\\; =\\; \\max_{\\pi}\\,
    \\mathbb{E}_{x\\sim D, y\\sim \\pi (y\\mid x)} \\bigl[\\, r (x, y)\\;
    -\\; \\log\\pi (y\\mid x)\\; +\\; \\log\\pi_{\\mathrm{ref}}(y\\mid
    x)\\bigr].$$\n\nSince $Z (x)$ does not depend on $\\pi$, the minimum is
    achieved by the policy that minimizes the first KL term. Gibbs'
    inequality tells us that the KL-divergence is minimized at 0 if and only
    if the two distributions are identical. Hence we have the optimal
    solution:\n$$\\pi^{*}(y\\mid x)\\; =\\; \\frac{1}{Z (x)}\\,
    \\pi_{\\mathrm{ref}}(y\\mid x)\\; \\exp\\Bigl (\\tfrac{1}{\\beta}\\, r
    (x, y)\\Bigr), $$\nfor all $x\\in\\mathcal{D}$. This completes the
    derivation.''\n\n\\(\\textit{Here, }\\; Z (x)=\\sum_{y}\\,
    \\pi_{\\mathrm{ref}}(y\\mid x)\\exp\\bigl (\\tfrac{1}{\\beta}r (x,
    y)\\bigr)\\; \\textit{ensures}\\; \\pi^{*}(y\\mid x)\\; \\textit{is a
    valid probability distribution.}\\)\n\nevidence: Appendix A.1, pp. 15-16
    of the paper"}
```

B.1.2 Gemini-2.0-Flash-Exp Results

Here we provide the results of Gemini-2.0-Flash-Exp as follows:

Gemini-2.0-Flash-Exp as Query Draft and Answer Retriever, using the same formula as Appendix B.1.1. It is evident that compared to Appendix B.1.1, the query generated by gemini-2.0-flash-exp did not fully display the formula conditions (formula 4 in Fig. 1), and the derivation process was overly general rather than a detailed step-by-step derivation.

formula:

```
"$$\\pi_{r}(y | x)=\\frac{1}{Z (x)} \\pi_{r e f}(y | x) \\exp
    (\\frac{1}{\\beta} r (x, y)), $$(4)"
```

query:

```
"Given the RL objective in Formula (3):$$\\max _{\\pi_{\\theta}}
    \\mathbb{E}_{x \\sim D, y \\sim \\pi_{\\theta}(y|x)}[r_{\\phi}(x,
    y)]-\\beta D_{K L}[\\pi_{\\theta}(y | x) || \\pi_{r e f}(y | x)], $$ how
    can we express the optimal solution $\\pi_{r}(y | x)$ to the
    KL-constrained reward maximization objective, where $Z (x)$ is the
    partition function?"
```

answer:

```
"Following prior work [31, 30, 19, 15], it is straightforward to show that
    the optimal solution to the KL-constrained reward maximization objective
    in Eq. 3 takes the form:\n$$\\pi_{r}(y | x)=\\frac{1}{Z (x)} \\pi_{r e
    f}(y | x) \\exp (\\frac{1}{\\beta} r (x, y)), $$(4)\nwhere $$Z (x) =
    \\sum_{y} \\pi_{r e f}(y | x) \\exp (\\frac{1}{\\beta} r (x, y)), $$ is
    the partition function. See Appendix A.1 for a complete derivation."
```

B.1.3 Gemini-2.0-Flash-Exp Results

Here we provide the results of Gemini-2.0-Flash-Exp as follows:

Gemini-2.0-Flash-Exp as Answer Retriever, using the same formula and query as Appendix B.1.1. It is evident that the result was less comprehensive and detailed than Appendix B.1.1.

formula:

```
"$$\\pi_{r}(y | x)=\\frac{1}{Z (x)} \\pi_{r e f}(y | x) \\exp
    (\\frac{1}{\\beta} r (x, y)), $$(4)"
```

query:

```
"Based on Formula (3): $$\\max _{\\pi_{\\theta}} \\mathbb{E}_{x \\sim D, y
    \\sim \\pi_{\\theta}(y|x)}[r_{\\phi}(x, y)]-\\beta D_{K
    L}[\\pi_{\\theta}(y | x) || \\pi_{r e f}(y | x)], $$ which enforces a
    KL-constrained reward maximization, how can we derive Formula (4): $$
    \\pi_{r}(y | x)=\\frac{1}{Z (x)} \\pi_{r e f}(y | x) \\exp\\Bigl
    (\\tfrac{1}{\\beta} r (x, y)\\Bigr), $$ where $Z (x)$ is the partition
    function ensuring $\\pi_{r}$ is a valid probability distribution?"
```

answer:

```
"Following prior work [31, 30, 19, 15], it is straightforward to show that
    the optimal solution to the KL-constrained reward maximization objective
    in Eq. 3 takes the form:\n$$\\pi_{r}(y | x)=\\frac{1}{Z (x)}
    \\pi_{\\textit{ref}}(y | x) \\exp (\\frac{1}{\\beta} r (x, y)), $$
    (4)\nwhere $Z (x)=\\sum_{y} \\pi_{\\textit{ref}}(y | x) \\exp
    (\\frac{1}{\\beta} r (x, y))$ is the partition function. evidence:
    Section A.1, Appendix"
```

B.2 Time Costs

On average, the processing time for each paper using the multi-agent system is approximately 20 minutes, with variability depending on the paper's length and the number of formulas. When utilizing GPT-o1-Pro for more challenging tasks, the processing time can be significantly longer. Moreover, output failures may occur, requiring multiple retries—sometimes two or even three times—leading to substantial time costs. Additionally, issues such as the "dumbing down" of LLMs during intensive tasks can further hinder experimental progress, creating delays in task completion. This represents a significant source of time cost, as repeated attempts are necessary to recover from failures and ensure valid results. In addition, there is no API for GPT-o1-Pro, so we have to use the web version. And the model can not receive pdf files, so we can only convert the paper into page screenshots and gradually upload, which increases the labor costs and time costs.

C Derivation Evaluation Prompt

We evaluated our method on a formula derivation test set using Deepseek-R1 with the prompt below to assess correctness, completeness, and similarity to ground-truth proofs (each scored on a 0–2 scale). We found that providing the ground-truth proof alongside the model's output was crucial for accurate comparison and reliable scoring.

Prompt of Answer Retriever

```
"""
You are a precise mathematical proof evaluator for proof problems. The user
    will provide both a problem statement, ground truth proof and a proposed
    solution.
    Evaluate the solution based on the ground truth proof, score the solution
    based on the following criteria:

    1. **Correctness (0-2):**
    - 0: Fundamentally wrong.
    - 1: Partially correct with significant flaws.
    - 2: Fully correct and logically sound.

    2. **Completeness (0-2):**
    - 0: Incomplete; key steps are missing.
    - 1: Moderately complete; some steps or justifications are missing.
    - 2: Fully complete; all necessary steps and justifications are present.

    3. **Similarity (0-2):**
    - 0: No similarity; completely different from the ground truth.
    - 1: Some similarity; some steps or justifications are similar.
    - 2: High similarity; all steps and justifications are identical.

    Output your evaluation as a JSON object in the format:
    {"correctness": <0-2>, "completeness": <0-2>, "similarity": <0-2>}
"""
```