

AceMath: Advancing Frontier Math Reasoning with Post-Training and Reward Modeling

Zihan Liu^{†*1}, Yang Chen^{†*1}, Mohammad Shoeybi¹, Bryan Catanzaro¹, Wei Ping^{†1}

¹NVIDIA

Abstract

In this paper, we introduce AceMath, a suite of frontier math models that excel in solving complex math problems, along with highly effective reward models capable of evaluating generated solutions and reliably identifying the correct ones. To develop the instruction-tuned math models, we propose a supervised fine-tuning (SFT) process that first achieves competitive performance across general domains, followed by targeted fine-tuning for the math domain using a carefully curated set of prompts and synthetically generated responses. The resulting model, *AceMath-72B-Instruct* greatly outperforms Qwen2.5-Math-72B-Instruct, GPT-4o and Claude-3.5 Sonnet. To develop math-specialized reward model, we first construct *AceMath-RewardBench*, a comprehensive and robust benchmark for evaluating math reward models across diverse problems and difficulty levels. After that, we present a systematic approach to build our math reward models. The resulting model, *AceMath-72B-RM*, consistently outperforms state-of-the-art reward models. Furthermore, when combining AceMath-72B-Instruct with AceMath-72B-RM, we achieve the highest average rm@8 score across the math reasoning benchmarks.

1 Introduction

Over the past year, the open large language model (LLM) community has made remarkable progress in advancing the key capabilities of LLMs, including multi-turn conversation (Chiang et al., 2023; Dubey et al., 2024), coding (Guo et al., 2024; Hui et al., 2024), multimodal functionalities (Dai et al., 2024; Chen et al., 2024), retrieval-augmented generation (RAG) (Liu et al., 2024c), and mathematical reasoning (Azerbayev et al., 2023; Shao et al., 2024; Mistral, 2024; Yang et al., 2024b).

Among these capabilities, mathematics is recognized as a fundamental aspect of intelligence. It can serve as a reliable benchmark due to its objective, consistent, verifiable nature. Consequently, solving math problems is widely regarded as a critical testbed for evaluating an LLM’s ability to tackle challenging tasks that require complex, numerical and multi-step logical reasoning (e.g., Hendrycks et al., 2021a; Lightman et al., 2023).

Previous studies have convincingly demonstrated that math-specialized LLMs significantly outperform general-purpose LLMs on challenging mathematical benchmarks (Shao et al., 2024; Mistral, 2024; Yang et al., 2024b). These math-specialized models, including the corresponding reward models (a.k.a. verifiers), are not only valuable to the mathematics and science communities (e.g., Tao, 2023), but they also provide valuable insights into data collection and serve as synthetic data generation tools, contributing to the advancement of future iterations of general-purpose LLMs.

In this work, we push the limits of math reasoning with post-training and reward modeling based on open weights *base* LLMs and *math base* LLMs. We establish state-of-the-art supervised fine-tuning (SFT) and reward modeling (RM) processes for building math-specialized models, while also sharing key insights gained from our comprehensive studies. Our contributions are as follows:

1. We introduce a SFT process designed to first achieve competitive performance across general domains, including multidisciplinary topics, coding, and math. Building on this, the general SFT model is further fine-tuned in math domain using a meticulously curated set of prompts and synthetically generated responses. Leveraging the high-quality training data, the resulting model, AceMath-7B-Instruct, largely outperforms the previous best-in-class Qwen2.5-Math-7B-Instruct (67.2 vs. 62.9) on a variety of math

*Equal contribution. † Correspondence to: Zihan Liu <zihanl@nvidia.com>, Yang Chen <yachen@nvidia.com>, Wei Ping <wping@nvidia.com>

	AceMath 72B-Instruct	AceMath 7B-Instruct	AceMath 1.5B-Instruct	Qwen2.5-Math 72B-Instruct	Qwen2.5-Math 7B-Instruct	Llama3.1 405B-Instruct	GPT-4o* 2024-08-06	Claude 3.5 Sonnet 2024-10-22
GSM8K Grade school math	96.4 97.1 (rm@8)	93.7 96.4 (rm@8)	87.0 93.9 (rm@8)	95.9 96.4 (rm@8)	95.2 97.9 (rm@8)	96.8	92.9	96.4
MATH High school math competition	86.1 89.4 (rm@8)	83.1 87.8 (rm@8)	76.8 84.9 (rm@8)	85.9 89.8 (rm@8)	83.6 88.5 (rm@8)	73.8	81.1	78.3
Minerva Math Undergraduate-level quantitative reasoning	57.0 59.9 (rm@8)	51.1 55.2 (rm@8)	41.5 49.3 (rm@8)	44.1 47.4 (rm@8)	37.1 42.6 (rm@8)	54.0	50.7	48.2
Gaokao 2023 English College-entry math exam	72.2 76.1 (rm@8)	68.1 76.1 (rm@8)	64.4 71.4 (rm@8)	71.9 76.9 (rm@8)	66.8 75.1 (rm@8)	62.1	67.5	64.9
Olympiad Bench Olympiad-level math reasoning	48.4 52.0 (rm@8)	42.2 50.2 (rm@8)	33.8 46.2 (rm@8)	49.0 54.5 (rm@8)	41.6 49.9 (rm@8)	34.8	43.3	37.9
College Math College-level mathematics	57.3 59.6 (rm@8)	56.6 59.5 (rm@8)	54.4 58.3 (rm@8)	49.5 50.6 (rm@8)	46.8 49.6 (rm@8)	49.3	48.5	48.5
MMLU STEM Undergraduate-level STEM knowledge	85.4 89.4 (rm@8)	75.3 86.0 (rm@8)	62.0 81.6 (rm@8)	80.8 80.1 (rm@8)	71.9 78.7 (rm@8)	83.1	87.9	85.1
Average	71.8 74.8 (rm@8)	67.2 73.0 (rm@8)	60.0 69.4 (rm@8)	68.2 70.8 (rm@8)	62.9 68.9 (rm@8)	64.8	67.4	65.6

* OpenAI's o1 model family is excluded in this table due to dependency on extensive pre-response computation mechanism.

Figure 1: AceMath *versus* leading open-weights and proprietary LLMs on math reasoning benchmarks. Additionally, we report rm@8 accuracy (best of 8) with our reward model AceMath-72B-RM and use the official reported numbers from Qwen2.5-Math.

reasoning benchmarks (see Figure 1), while coming close to the performance of 10× larger Qwen2.5-Math-72B-Instruct (67.2 vs. 68.2).

2. We conducted a systematic investigation of training techniques for building math-specialized reward models, focusing on key aspects such as the construction of positive-negative pairs, training objectives, and the elimination of stylistic biases. Leveraging the insights gained from this exploration, our AceMath-72B-RM consistently outperforms state-of-the-art reward models, including Qwen2.5-Math-RM-72B and Skywork-o1-Open-PRM-Qwen-2.5-7B (Skywork-o1, 2024), in the math domain. Moreover, when combining AceMath-72B-Instruct with AceMath-72B-RM, we achieve the highest average rm@8 score across seven math reasoning benchmarks (see Figure 1), setting a new standard for performance in this field.
3. We open source the model weights for AceMath-Instruct and AceMath-RM, along with the complete training data used across all stages of their development. We also release AceMath-RewardBench, a comprehensive and diverse benchmark for evaluating math reward models.

2 Related Work

2.1 Math Post-Training

Math-instructed models have been developed to advance LLM performance in the mathematics do-

main (Shao et al., 2024; Toshniwal et al., 2024; Yang et al., 2024b; Muennighoff et al., 2025) by utilizing math-specific pretrained models as backbones and vast amounts of synthetic post-training data tailored to mathematics. Reinforcement learning (RL) has recently emerged as a pivotal technique in enhancing the math reasoning performance, which allows models to iteratively refine their outputs based on the correctness of generated solutions (Team et al., 2025; Guo et al., 2025).

2.2 Math Reward Modeling

Generative outcome reward models, such as LLM-as-a-judge (Zheng et al., 2023) prompt LLMs to act as verifiers using predefined rubrics and grading templates (Bai et al., 2022; Zhang et al., 2024c; Yu et al., 2024). In contrast, process reward models (PRMs) provide step-by-step evaluations of model responses (Uesato et al., 2022; Lightman et al., 2023). For example, Math-Shepherd (Wang et al., 2024b) introduces an automated sampling method to construct large-scale process supervision data for training, following by further developments in step-wise supervision labeling (Dong et al., 2024), including PAV (Setlur et al., 2024), OmegaPRM (Luo et al., 2024), ER-PRM (Zhang et al., 2024b), AutoPSV (Lu et al., 2024) and ProcessBench (Zheng et al., 2024). More details on related work can be found in Appendix C.

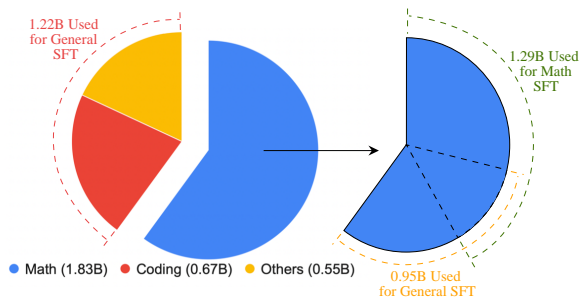


Figure 2: The proportion of total SFT tokens for math, coding, and other categories.

3 Supervised Fine-tuning

3.1 Overview

Providing a strong initialization point is crucial for the model to begin math-focused SFT effectively. Previous works (Shao et al., 2024; Yang et al., 2024b) have demonstrated that continual pre-training of LLMs with a large math corpus provides a more effective initialization for subsequent math post-training. Taking this further, we explore whether conducting general SFT on pre-trained LLM can serve as an even better initialization for the subsequent math-specific SFT. The idea is that performing SFT on general-purpose tasks helps the model develop strong capabilities for following instructions and reasoning (e.g., knowledge-related). This foundation, in turn, makes it easier for the model to acquire math problem-solving skills from math-focused SFT data. We call the resulting general SFT model “AceInstruct”. The details of curating general SFT data can be found in §3.2.1.

The next-step is constructing math-specific SFT data. It is crucial to develop a diverse set of math prompts accompanied by unified, step-by-step, and accurate solutions. The details of curating math SFT data can be found in §3.2.2.

Figure 2 depicts the summary of the SFT data. The details of how we leverage general and math SFT data for the training can be found in §3.3.

3.2 Data Curation

3.2.1 General SFT Data

Our goal is to build a general SFT model that serves as a strong starting point for the subsequent math-specific SFT. This general SFT model should excel at following instructions and answer a wide range of questions (e.g., math and coding).

Prompt Construction To achieve this goal, we collect prompts from a diverse range of open-

source datasets, categorized as follows:

- ▷ General domain: ShareGPT (Chiang et al., 2023; The-Vicuna-Team, 2023), SlimOrca (Lian et al., 2023; Mukherjee et al., 2023), EvoInstruct (Xu et al., 2024), GPTeacher (Teknium, 2023), AlpacaGPT4 (Peng et al., 2023), and UltraInteract (Yuan et al., 2024);

- ▷ Coding domain: Magicoder (Wei et al., 2024), WizardCoder (Luo et al., 2023), GlaiveCodeAssistant (Glaive-AI, 2023), and CodeSFT (Adler et al., 2024);

- ▷ Math domain: NuminaMath (Li et al., 2024b), OrcaMathWordProblems (Mitra et al., 2024), MathInstruct (Yue et al., 2024a), MetaMathQA (Yu et al., 2023), and our synthetic data (details in §3.2.2).

Since different data sources could have prompt overlaps, we conduct data deduplication to eliminate duplicate prompts that are identical when converted to lowercase. After deduplication, we retain the prompt set unfiltered to preserve the diversity of prompts.

Response Construction After collecting the prompts, our goal is to construct high-quality responses in a consistent format so that models can learn more effectively. Therefore, we avoid using the original open-source responses for these prompts, as they may lack quality and have inconsistent formats due to being sourced from different curators or generated by different models. We use GPT-4o-mini (2024-0718) to generate responses for collected prompts in coding and general domains. GPT-4o-mini is selected for its strong performance across different tasks and instructions, as well as its compact size, which makes it both time-efficient and cost-efficient for producing a large volume of generated responses. Details of getting responses for math SFT prompts are in §3.2.2.

We generate a single response for each prompt using greedy decoding, ultimately accumulating around 1.2 million coding SFT samples (0.67 billion tokens) and 0.7 million samples (0.55 billion tokens) in the general domain. And, we take around 1.2 million samples (0.95 billion tokens) from the math SFT data for conducting general SFT.

3.2.2 Math SFT Data

The goal is to construct a diverse set of math prompts accompanied by unified, step-by-step, and accurate solutions.

Initial Prompts We first take math prompts from general SFT data, drawing specifically from open-

source datasets: NuminaMath (Li et al., 2024b), OrcaMathWordProblems (Mitra et al., 2024), Math-Instruct (Yue et al., 2024a), and MetaMathQA (Yu et al., 2023). These prompts cover a wide range of math problems, spanning grade school, high school, college-level, and Olympiad-level math challenges. After that, we perform data deduplication to remove duplicate prompts as before. Finally, we collect over 1.3 million initial prompts.

Synthetic Prompt Generation Furthermore, we generate additional synthetic prompts to enrich the diversity of our math prompt collection. We select NuminaMath as our seed prompt source due to its broad coverage of math questions across various difficulty levels. Then, we apply the strategies inspired by Xu et al. (2024) and construct another 1 million synthetic prompts. More details can be found in Appendix D.1.

Response Construction We utilize Qwen2.5-Math-72B-Instruct for generating responses to math prompts, given its remarkable performance across various math benchmarks. Eventually, we obtain a total of around 2.3 million math SFT samples (1.83 billion tokens), of which around 1.2 million are utilized in the general SFT. Details of the response construction and the data filtering process can be found in Appendix D.2. In addition, we discuss data contamination details in Appendix D.3.

3.3 Training Strategy

3.3.1 General SFT Strategy

Among general tasks, solving complex coding and math problems stands out as particularly challenging, and many general instruct models often struggle with them. To address this and develop a more effective general SFT model, we introduce a two-stage training approach. In stage-1, the model is trained on a large dataset specifically curated for code and math SFT tasks, providing a strong foundation in these areas. Stage-2 expands the scope by incorporating a balanced mix of code, math, and other general SFT data, broadening the model’s capabilities and enhance the overall performance. See details in Appendix D.4.

3.3.2 Math SFT Strategy

We take the base (or math-base) model trained on our general SFT data as the starting point for the math SFT. In order to achieve diverse and high-quality math SFT data, we merge all samples from

NuminaMath (Li et al., 2024b), a subset of samples from our synthetic prompts, and the 800K math SFT samples that are cross-checked between GPT-4o-mini and Qwen2.5-Math-72B-Instruct (as described in §3.2.2). We remove duplicate samples with identical prompts, resulting in a total of 1.6 million samples for math SFT. We find that this training blend leads to better results than directly utilize all 2.3 million math SFT samples for training (this ablation study can be found in §3.6.3).

3.3.3 SFT Data Summary

Figure 2 provides an overview of the distribution of total SFT tokens across math, coding, and other categories, along with details on the utilization of math SFT samples. In total, there are approximately 2.3 million math SFT samples (1.83 billion tokens), 1.2 million coding SFT samples (0.67 billion tokens), and 0.7 million samples in other categories (0.55 billion tokens). Among the math SFT samples, 1.2 million (0.95 billion tokens) are used for general SFT, while 1.6 million (1.29 billion tokens) are utilized for math SFT. The SFT training hyperparameters are in Appendix D.5.

3.4 Benchmarks

3.4.1 General SFT Benchmarks

We evaluate our general SFT models on a diverse set of widely used benchmarks. These benchmarks consist of coding tasks, including HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021), mathematical reasoning, including GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021b), as well as general knowledge domains, including MMLU (Hendrycks et al., 2020) and MMLU Pro (Wang et al., 2024c).

3.4.2 Mathematical Benchmarks

We follow the evaluation setting in Qwen2.5-Math (Yang et al., 2024b) for assessing English mathematical tasks. Beyond the commonly used GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021b) benchmarks, we also evaluate our models on a broader set of mathematical benchmarks, including Minerva Math (Lewkowycz et al., 2022), GaoKao 2023 En (Liao et al., 2024), Olympiad Bench (He et al., 2024), College Math (Tang et al., 2024), and MMLU STEM (Hendrycks et al., 2020). These benchmarks comprehensively assess a wide range of mathematical reasoning capabilities, from grade

Models	HumanEval	MBPP	GSM8K	MATH	MMLU	MMLU Pro	Avg.
DeepSeek-Coder-7B-Instruct-v1.5	64.10	64.60	72.60	34.10	49.50	-	-
AceInstruct-7B-DeepSeekCoder (Ours)	78.05	73.54	82.56	55.62	54.65	33.28	62.95
Llama3.1-8B-Instruct	72.60	69.60	84.50	51.90	69.40	48.30	66.05
AceInstruct-8B-Llama3.1 (Ours)	81.10	74.71	90.45	64.42	68.31	43.27	70.38
Qwen2.5-1.5B-Instruct	61.60	63.20	73.20	55.20	58.37	32.40	57.33
AceInstruct-1.5B-Qwen2.5 (Ours)	73.17	65.76	80.44	60.34	58.17	33.78	61.94
Qwen2.5-7B-Instruct	84.80	79.20	91.60	75.50	74.51	56.30	76.99
AceInstruct-7B-Qwen2.5 (Ours)	85.37	74.32	93.10	76.40	74.68	54.50	76.40
Qwen2.5-72B-Instruct	86.60	88.20	95.80	83.10	84.67	71.10	84.91
AceInstruct-72B-Qwen2.5 (Ours)	89.63	83.66	96.36	84.50	83.88	66.10	84.02

Table 1: Results of our AceInstruct general SFT models. We apply our proposed two-stage training strategy to conduct SFT on various base models from DeepSeekCoder, Llama3.1, and Qwen2.5. These finetuned models are then compared against the corresponding instruct baselines that are built upon the same base models.

school arithmetic to advanced college-level problems and Olympic-level challenges. For all benchmarks except for Math and GSM8K, we do not use any training dataset or synthetic dataset derived from it. This ensures a more reliable and valid evaluation of our models on these benchmarks.

3.5 AceInstruct Results

As shown in Table 1, we apply our proposed two-stage training strategy to conduct SFT on various base models, including DeepSeekCoder-7B (Guo et al., 2024), Llama3.1-8B (Dubey et al., 2024), and Qwen2.5-1.5B/7B/72B (Yang et al., 2024a). We compare our finetuned general AceInstruct models to the corresponding instruct baselines that are built upon the same base models. We observe that our general SFT brings significant improvements across different models, such as DeepSeek-Coder-7B, Llama3.1-8B, and Qwen2.5-1.5B, with an average score improvement of over 4%. Notably, results on DeepSeek-Coder show that AceInstruct achieves particularly pronounced gains, with an average score increase of approximately 10% or more in coding and math tasks. When compared to more advanced models like Qwen2.5-7B-Instruct and Qwen2.5-72B-instruct, our SFT delivers comparable performance. These findings highlight the effectiveness and strong generalization capabilities of our constructed general SFT dataset.

We also study the effectiveness of two-stage training strategy for AceInstruct. The results can be found in Appendix A.1.

3.6 AceMath-Instruct Results

3.6.1 Main Results

In Figure 1, we compare our AceMath-Instruct models against several strong baselines for greedy

decoding, including Qwen2.5-Math-7B/72B-Instruct (Yang et al., 2024b), GPT-4o (OpenAI, 2024b), Llama3.1-405B-Instruct, and Claude-3.5 Sonnet (Anthropic, 2024). Specifically, our AceMath-1.5B/7B/72B-Instruct models are built upon the Qwen2.5-Math-1.5B/7B/72B-base models, which also serve as the foundation for Qwen2.5-Math-1.5B/7B/72B-Instruct. We find that AceMath-1.5B/7B/72B-Instruct achieve significantly better performance compared to the corresponding Qwen2.5-Math-1.5B/7B/72B-Instruct models. Our best model, AceMath-72B-Instruct, achieves a significant average improvement of 3.68 over the previous state-of-the-art, Qwen2.5-Math-72B-Instruct. This highlights the superior quality and generalizability of our math SFT data.

Moreover, we find that our 7B model, AceMath-7B-Instruct, demonstrate superior or comparable performance compared to several advanced instruct models, including Llama3.1-405B-Instruct, GPT-4o, and Claude-3.5 Sonnet. And, it comes close to matching the performance of the significantly larger Qwen2.5-Math-72B-Instruct, with only a slight difference in the average score (68.16 vs. 67.17). We put several chain-of-thought reasoning examples generated by AceMath-72B-Instruct in Appendix E.

3.6.2 Backbone Model: Base vs. Math-Base

In Figure 3, we study the impact of using either the *base* model (e.g., Qwen2.5-7B-Base) or the *math base* model (e.g., Qwen2.5-Math-7B-Base) as the backbone on the performance of our AceMath-Instruct models. This study is crucial, as it helps us understand the importance of continual pre-training on a large math corpus (i.e., building *math base* models) for improving the performance on solving

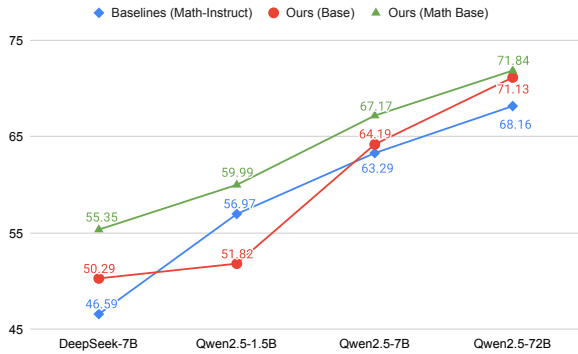


Figure 3: Studies on the impact of using either the base model or the math base model as the backbone on the performance of our AceMath-Instruct models. We compare our models against the corresponding math-instruct baselines across different model types and sizes. Results are the average scores of greedy decoding over the math benchmarks.

math questions after post-training.

For DeepSeek-7B, “Ours (Base)” uses the DeepSeek-Coder-7B-Base (Guo et al., 2024) as the backbone model, while “Ours (Math Base)” uses the DeepSeek-Math-7B-Base (Shao et al., 2024) as the backbone model, which continues the pre-training of DeepSeek-Coder-7B-Base using a large math corpus. The math instruct baseline is DeepSeek-Math-7B-RL (Shao et al., 2024), which is developed from DeepSeek-Math-7B-Base. For Qwen2.5-1.5/7B/72B, the base models are Qwen2.5-1.5/7B/72B-Base, while the math base models are Qwen2.5-Math-1.5/7B/72B-Base, with the baselines being Qwen2.5-Math-1.5/7B/72B-Instruct.

We find that as the model size increases, the performance of our models with base models as backbones approaches that of models with math base as backbones. Specifically, when the Qwen2.5-(Math)-72B-Base is used, the performance gap between “Ours (Base)” and “Ours (Math Base)” becomes very marginal (71.84 vs. 71.13). We conjecture that larger models inherently possess better math problem-solving and generalization capability, which diminishes the need for continual pre-training. This finding extends across different model families. Additionally, when comparing models of sizes between 1.5B and 7B, the performance gap between “Ours (Base)” and “Ours (Math Base)” is smaller for 7B models (i.e., DeepSeek-7B and Qwen2.5-7B) than it is for Qwen2.5-1.5B. We put more details in Appendix A.2.

Models	Average
AceMath-Instruct	64.19
▷ Removing all synthetic data	62.53
▷ Using extra low-quality synthetic data	62.95

Table 2: Ablation studies on the synthetic data, exploring the effects of removing all synthetic math SFT data and incorporating additional low-quality synthetic math SFT data. The backbone of AceMath-Instruct is Qwen2.5-7B-Base. Results are average across the seven math benchmark.

3.6.3 Ablation Studies on Training Data

As shown in Table 2, we study how synthetic math SFT data affects the results. We compare AceMath-Instruct against two scenarios: one where all one million synthetic data samples are removed and another where an additional 500K low-quality synthetic data are included for training (e.g., lengthy prompts and one type of *in-depth evolution* that adds constraints). Details of the synthetic math SFT data can be found in §3.2.2. In both scenarios, we observe a decline in results, underscoring the importance of not only generating synthetic data but also carefully selecting it for training. Effectively leveraging appropriate synthetic data proves essential for achieving optimal performance.

Further more, we conduct ablation studies on more training data (e.g., math SFT samples) and strategies (e.g., the impact of conducting general SFT) across various backbone models for training our AceMath-Instruct models. We put all the details in Appendix A.3.

4 Reward Model Training

We train a math reward model for AceMath-Instruct, aiming to select more accurate solutions and better reasoning paths. To ensure broad applicability across a variety of language models, we curate a diverse training dataset. The following sections detail our training methodology, evaluation protocols, and empirical results.

4.1 Reward Training Data Synthesis

4.1.1 Initial Dataset Construction

We utilize a portion of the math SFT dataset (350K) from §3.2.2 to use the prompts and the answers generated by gpt-4o-mini (OpenAI, 2024a) as reference labels. To capture the diversity of model-generated reasoning steps and potential different kinds of reasoning mistakes, we sam-

ple four model responses per LLM from a set of 14 LLMs, including Llama2-7b-chat (Touvron et al., 2023), Llama3.1-8/70B-Instruct (Dubey et al., 2024), DeepSeek-math-7b-instruct (Shao et al., 2024), Mistral-7B/Mathstral-7B (Jiang et al., 2023), Gemma-2/27b-it (Gemma et al., 2024), and Qwen2/2.5-1.5/7/72B-Instruct (Yang et al., 2024b). We then annotate the model solutions as correct or incorrect by comparing them against the referenced labels using the Qwen-math evaluation toolkit.¹ This process initializes a pool of correct and incorrect candidate responses for each problem, which we treat as positive and negative samples that can be further sampled to create paired responses for training.

4.1.2 Response Scoring and Selection

Mathematical problem answers encompass a wide range of formats with diverse representations (e.g., $[\frac{1}{2}]$, $1/2$, 0.5) and $[1e-5, 1 \times 10^{-5}]$, and heuristic math evaluation toolkits using SymPy and latex2sympy2 may inevitably result in false negative candidates (i.e., correct answers annotated as incorrect). Such examples in the negative candidates could introduce noise and adversely affect model training. Therefore, instead of randomly sample responses from all candidates, we rank the candidates and apply a score-sorted sampling strategy. Specifically, we use the math reward model Qwen2.5-Math-RM-72B to rank positive and negative candidates for each problem based on their scores. We then randomly sample from top- k positive and bottom- k negative candidates, with k set to 14 based on preliminary experiments. In conclusion, we sample a total of six response candidates (positive + negative) for each problem, ensuring a balanced number of positive and negative responses, and filter out problems where all responses are either correct or incorrect.

4.1.3 Addressing Stylistic Biases

LLMs can generate different styles of chain-of-thought reasoning paths when prompted in the zero-shot setting or with few-shot examples (Wei et al., 2022). We observe significant shorter and simple reasoning paths in model outputs for datasets such as MMLU (Hendrycks et al., 2021a) as the model follows the simple 5-shot examples provided in the instruction. To improve reward model performance on such output styles, we create training data us-

ing the few-shot prompting approach to generate simple and short reasoning paths for 2,000 multiple-choice problems. In addition, as our ultimate goal is to develop a reward model for the AceMath-Instruct model family, we sample a set of 30,000 problems and use AceMath-(1.5/7/72B)-Instruct checkpoints to generated responses to create positive and negative pairs for training. In conclusion, our final training dataset consists of 356K problems, each paired with a total of six responses (k positive and $6 - k$ negative).

4.2 Reward Training Strategy

Our reward model architecture adopts a outcome reward approach, which introduces a linear layer at the top of the language model to project the last token representation into a scalar value. We initialize the backbone of the reward model using AceMath-Instruct. Following the training objective in Qwen2.5-Math (Yang et al., 2024b), we construct problem-response pairs with k positive (correct) candidates and $6 - k$ negative (incorrect) candidates. We compute the list-wise Bradley-Terry loss (Bradley and Terry, 1952), which demonstrates computational efficiency compared to pair-wise approaches as shown in Table 9.

$$\mathcal{L}_m(\theta) = -\frac{1}{k \cdot (6 - k)} \mathbb{E}_{(x, y_{\text{pos}}, y_{\text{neg}})} \left[\log(\sigma(r_\theta(x, y_{\text{pos}}) - r_\theta(x, y_{\text{neg}}))) \right]$$

Here, $r_\theta(x, y)$ represents the output score of the reward model r_θ , where x denotes the problem and y represents the response candidate.

4.3 Reward Evaluation Benchmarks

4.3.1 AceMath-RewardBench

Existing math reward benchmarks lack diversity, both in the types of candidate solutions and the range of difficulty levels in the math questions. To address this, we construct a math reward model evaluation benchmark, AceMath-RewardBench, which contains 7 datasets and use 8 different LLMs to generate solutions for robust evaluation. The benchmark use the best-of- N (BoN or $\text{rm}@n$) metric, a methodology extensively used in literature (Cobbe et al., 2021; Lightman et al., 2023; Yang et al., 2024b). The primary objective of the reward model is to select the highest reward scored model response from a candidate set of n and calculate the corresponding problem-solving rate for each math benchmark (7 datasets) used in §3.4.2.

¹<https://github.com/QwenLM/Qwen2.5-Math/tree/main/evaluation>

Model	GSM8K	MATH500	Minerva Math	GaoKao 2023 En	Olympiad Bench	College Math	MMLU STEM	Avg.
majority@8	96.22	83.11	41.20	68.21	42.69	45.01	78.21	64.95
Internlm2-7b-reward	95.26	78.96	36.25	67.51	40.49	43.88	75.42	62.54
Internlm2-20b-reward	95.10	76.53	37.69	66.63	40.12	42.57	70.60	61.32
Skywork-Reward-Llama-3.1-8B-v0.2	95.64	74.16	39.11	67.16	39.10	44.58	76.52	62.32
Skywork-Reward-Gemma-2-27B-v0.2	95.94	74.90	39.37	66.96	39.07	45.46	78.20	62.84
Skywork-o1-Open-PRM-Qwen-2.5-7B	96.92	86.64	41.00	72.34	46.50	46.30	74.01	66.24
Qwen2.5-Math-RM-72B	96.61	86.63	43.60	73.62	47.21	47.29	84.24	68.46
AceMath-7B-RM (Ours)	96.66	85.47	41.96	73.82	46.81	46.37	80.78	67.41
AceMath-72B-RM (Ours)	97.23	86.72	45.06	74.69	49.23	46.79	87.01	69.53
pass@8 (Oracle)	98.86	91.84	56.18	82.09	59.00	56.38	96.15	77.21

Table 3: **Reward model evaluation on AceMath-RewardBench.** The average results (rm@8) of reward models on math benchmarks, randomly sample 8 responses from 64 candidates with 100 random seeds. Response candidates are generated from a pool of 8 LLMs (Qwen{2/2.5}-Math-{7/72}B-Instruct, Llama-3.1-{8/70}B-Instruct, Mathtral-7B-v0.1, deepseek-math-7b-instruct).

We adopt rm@8 metric following the Qwen2.5-Math evaluation protocol, optimizing computational efficiency during the inference stage. To ensure robust and statistically reliable benchmark performance, we implement two design principles: 1) diverse model distribution: we sample 8 responses from each model in a set of mathematical and general-purpose LLMs (see Table 3), mitigating potential model-specific style biases; 2) we compute accuracy metrics by averaging results across 100 random seeds, reducing result variance and enhancing reproducibility. In total, each problem in the benchmark contains a total of 64 candidate responses from 8 LLMs. We then randomly sample 8 responses from these 64 candidates, compute the rm@8 result, and average the final accuracy over 100 random seeds.

4.3.2 RewardBench and RewardMath

Apart from our own benchmarks, we also evaluate on RewardBench (Lambert et al., 2024) (MATH500) and RewardMath (Kim et al., 2024) to report the accuracy of selecting the correct solution from a list of candidates for each problem in MATH500 (Lightman et al., 2023). The primary difference between these two benchmarks lies in the candidate sets: RewardBench uses one correct (human-written) solution and one incorrect candidate (generated by GPT-4), while RewardMath uses one correct (a GPT-4 rewrite) and nine incorrect candidates (generated by models).

4.4 Experiments of Reward models

4.4.1 Hyperparameters

We use the AceMath-7B/72B-Instruct model as the backbone to train the outcome reward model: AceMath-RM-7/72B. The model is trained using AdamW (Kingma, 2014; Loshchilov, 2017) for 2 epochs with a learning rate of {5e-6, 2e-6}, using a cosine learning rate scheduler and an effective batch size of 256.

4.4.2 Baselines

For mathematical reward modeling, we compare with current state-of-the-art outcome reward model Qwen2.5-Math-RM-72B (Yang et al., 2024b) and a process reward model Skywork-o1-Open-PRM-Qwen-2.5-7B (Skywork-o1, 2024). We also include majority@8 (majority voting) baseline and the pass@8 (any one of the 8 is correct) as an oracle reward model to measure the upper bound of this benchmark. Additionally, we incorporate general reward models top-ranked on RewardBench, including Skywork-Reward (Liu et al., 2024a) and Internlm2-reward (Cai et al., 2024). It is noteworthy that while these models are not exclusively trained for mathematical domains, a substantial portion of their training data encompasses mathematical content.

4.4.3 AceMath-RewardBench Results

In Table 3, we show that our AceMath-72B-RM achieves the state-of-the-art rm@8 accuracy on average of AceMath-RewardBench, outperforming the Qwen2.5-Math-RM-72B by 1% absolute (69.53 vs 68.46) and on 6 out of 7 datasets. We show the 7B variant achieves 67.41 accuracy on average

Model	RewardBench	RewardMath
	MATH500	MATH500
<i>Random</i>	50.00	10.00
<i>LLM-as-a-Judge</i>		
Claude-3.5-Sonnet [†]	70.70	15.32
GPT-4o-2024-05-13 [†]	72.50	25.98
<i>Classifier-based</i>		
Math-Shepherd-Mistral-7B [†]	94.41	17.18
ArmoRM-Llama3-8B-v0.1 [†]	98.70	20.50
Skywork-Reward-Llama-3.1-8B [†]	96.87	22.15
Internlm2-20b-reward [†]	95.10	33.95
Internlm2-7b-reward [†]	94.90	37.27
Skywork-o1-Open-PRM-7B	78.52	51.34
Qwen2.5-Math-RM-72B	95.97	<u>68.53</u>
AceMath-7B-RM (Ours)	92.62	57.76
AceMath-72B-RM (Ours)	<u>97.09</u>	68.94

Table 4: The accuracy of reward models on RewardBench (MATH500) (Lambert et al., 2024) and RewardMATH (Kim et al., 2024). †: Results are copied from RewardMATH. **Bold**: top-1. Underline: top-2 accuracy.

and demonstrates the benefits of model size scaling from 7B to 72B, especially on datasets require college-level STEM knowledge such as Minerva Math (41.96 \rightarrow 45.06) and MMLU STEM (80.78 \rightarrow 87.01). Comparing to other reward model baselines, the 7B outperform Internlm2 and Skywork-Reward by a large margin as our benchmark reveal these reward model even underperform the majority voting baseline. Nevertheless, we note that there remains considerable room for improvement as indicated by the gap between the reward model and pass@8 oracle accuracy.

4.4.4 RewardBench & RewardMath Results

In Table 4, we demonstrate that our AceMath-72B-RM achieves state-of-the-art accuracy on RewardMATH. While many reward models (e.g., ArmoRM (Wang et al., 2024a), Internlm2) achieve 95%+ accuracy on the RewardBench MATH500 split, their accuracy drops significantly on RewardMATH, ranging from only 20% to 37%. We found Skywork-PRM model performs much better on RewardMATH (51.34) but worse on RewardBench (78.5). This may be due to the lack of reasoning steps typically found in human-written solutions, and as a result, our AceMath-7B-RM outperforms it on both benchmarks. In conclusion, these evaluation results highlight the benefits of training on diverse, model-generated solutions to mitigate, though not entirely eliminate, out-of-distribution generalization challenges.

4.4.5 Ablation Studies & rm@k

Due to the space limit, we present ablation studies on model backbone, data sampling, and loss functions for training the reward model in Appendix B.1. In addition, a comparison between AceMath-72B-RM and Qwen2.5-Math-RM-72B on rm@k can be found in Appendix B.2.

5 Conclusion

In this work, we present AceMath, a series of frontier-class math instruct and reward models. We demonstrate that our AceMath-7B-Instruct significantly surpasses the previous best-in-class Qwen2.5-Math-7B-Instruct across comprehensive math reasoning benchmarks, and it performs slightly worse than a 10 \times larger Qwen2.5-Math-72-Instruct (67.2 vs. 68.2). Remarkably, our AceMath-72B-Instruct greatly outperforms Qwen2.5-Math-72-Instruct, GPT-4o and Claude-3.5 Sonnet. Additionally, we construct AceMath-RewardBench, a comprehensive benchmark designed to evaluate math reward models across a diverse range of datasets and difficulty levels. We show that our AceMath-72B-RM consistently outperforms state-of-the-art reward models, including Qwen2.5-Math-RM-72B on various math reward benchmarks. Furthermore, when combining AceMath-72B-Instruct with AceMath-72B-RM, we achieve the highest average rm@8 score across the math reasoning benchmarks. To advance open research in the field, we will open source both AceMath-Instruct and AceMath-RM, along with the complete training data used throughout their development.

Limitation

AceMath models are specialized in math problems but not for other domains. Additionally, AceMath-Instruct models are trained using short Chain-of-Thought (CoT) datasets, limiting their ability to perform long Chain-of-Thought reasoning. This constraint reduces their effectiveness on highly complex math problems that require extensive reasoning. To overcome these limitations, our future work introduces AceReason-Nemotron (Chen et al., 2025), which leverages reinforcement learning (RL) on a SFT model to significantly enhance its long CoT reasoning capabilities in both mathematical and coding tasks.

References

- Bo Adler, Niket Agarwal, Ashwath Aithal, Dong H Anh, Pallab Bhattacharya, Annika Brundyn, Jared Casper, Bryan Catanzaro, Sharon Clay, Jonathan Cohen, et al. 2024. Nemotron-4 340b technical report. *arXiv preprint arXiv:2406.11704*.
- Syeda Nahida Akter, Shrimai Prabhumoye, John Kamalu, Sanjeev Satheesh, Eric Nyberg, Mostofa Patwary, Mohammad Shoeybi, and Bryan Catanzaro. 2024. Mind: Math informed synthetic dialogues for pretraining llms. *arXiv preprint arXiv:2410.12881*.
- Anthropic. 2024. [Introducing claude 3.5 sonnet](#).
- Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. 2021. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.
- Zhangir Azerbayev, Hailey Schoelkopf, Keiran Paster, Marco Dos Santos, Stephen McAleer, Albert Q Jiang, Jia Deng, Stella Biderman, and Sean Welleck. 2023. Llemma: An open language model for mathematics. *arXiv preprint arXiv:2310.10631*.
- Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. 2022. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*.
- Ralph Allan Bradley and Milton E. Terry. 1952. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345.
- Zheng Cai, Maosong Cao, Haojong Chen, Kai Chen, Keyu Chen, Xin Chen, Xun Chen, Zehui Chen, Zhi Chen, Pei Chu, et al. 2024. Internlm2 technical report. *arXiv preprint arXiv:2403.17297*.
- Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.
- Yang Chen, Zhuolin Yang, Zihan Liu, Chankyu Lee, Peng Xu, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. 2025. Acereason-nemotron: Advancing math and code reasoning through reinforcement learning. *arXiv preprint arXiv:2505.16400*.
- Zhe Chen, Weiyun Wang, Hao Tian, Shenglong Ye, Zhangwei Gao, Erfei Cui, Wenwen Tong, Kongzhi Hu, Jiapeng Luo, Zheng Ma, et al. 2024. How far are we to gpt-4v? closing the gap to commercial multimodal models with open-source suites. *arXiv preprint arXiv:2404.16821*.
- Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.
- Wenliang Dai, Nayeon Lee, Boxin Wang, Zhuolin Yang, Zihan Liu, Jon Barker, Tuomas Rintamaki, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. 2024. NVLM: Open frontier-class multimodal LLMs. *arXiv preprint arXiv:2409.11402*.
- Hanze Dong, Wei Xiong, Bo Pang, Haoxiang Wang, Han Zhao, Yingbo Zhou, Nan Jiang, Doyen Sahoo, Caiming Xiong, and Tong Zhang. 2024. Rlhf workflow: From reward modeling to online rlhf. *TMLR*.
- Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.
- Gemma, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. 2024. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*.
- Glaive-AI. 2023. [GlaiveCodeAssistant](#).
- Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.
- Daya Guo, Qihao Zhu, Dejian Yang, Zhenda Xie, Kai Dong, Wentao Zhang, Guanting Chen, Xiao Bi, YK Li, et al. 2024. Deepseek-coder: When the large language model meets programming—the rise of code intelligence. *arXiv preprint arXiv:2401.14196*.
- Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, et al. 2024. Olympiad-bench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. *arXiv preprint arXiv:2402.14008*.
- Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. In *International Conference on Learning Representations*.

- Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021a. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.
- Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021b. Measuring mathematical problem solving with the math dataset. *NeurIPS*.
- Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, et al. 2024. Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186*.
- Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.
- Sunghwan Kim, Dongjin Kang, Taeyoon Kwon, Hyungjoo Chae, Jungsoo Won, Dongha Lee, and Jinyoung Yeo. 2024. Evaluating robustness of reward models for mathematical reasoning. *arXiv preprint arXiv:2410.01729*.
- Diederik P Kingma. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.
- Nathan Lambert, Valentina Pyatkin, Jacob Morrison, LJ Miranda, Bill Yuchen Lin, Khyathi Chandu, Nouha Dziri, Sachin Kumar, Tom Zick, Yejin Choi, et al. 2024. Rewardbench: Evaluating reward models for language modeling. *arXiv preprint arXiv:2403.13787*.
- Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. 2022. Solving quantitative reasoning problems with language models. *Advances in Neural Information Processing Systems*, 35:3843–3857.
- Chengpeng Li, Zheng Yuan, Hongyi Yuan, Guanting Dong, Keming Lu, Jiancan Wu, Chuanqi Tan, Xiang Wang, and Chang Zhou. 2024a. Mugglemath: Assessing the impact of query and response augmentation on math reasoning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10230–10258.
- 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. 2024b. NuminaMath. [<https://huggingface.co/AI-MO/NuminaMath-CoT>](https://github.com/project-numina/aimo-progress-prize/blob/main/report/numina_dataset.pdf).
- Wing Lian, Guan Wang, Bleys Goodson, Eugene Pentland, Austin Cook, Chanvichet Vong, and "Teknium". 2023. Slimorca: An open dataset of gpt-4 augmented flan reasoning traces, with verification.
- Minpeng Liao, Wei Luo, Chengxi Li, Jing Wu, and Kai Fan. 2024. Mario: Math reasoning with code interpreter output—a reproducible pipeline. *arXiv preprint arXiv:2401.08190*.
- Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let’s verify step by step. *arXiv preprint arXiv:2305.20050*.
- Chris Yuhao Liu, Liang Zeng, Jiakai Liu, Rui Yan, Julie He, Chaojie Wang, Shuicheng Yan, Yang Liu, and Yahui Zhou. 2024a. Skywork-reward: Bag of tricks for reward modeling in llms. *arXiv preprint arXiv:2410.18451*.
- Yantao Liu, Zijun Yao, Rui Min, Yixin Cao, Lei Hou, and Juanzi Li. 2024b. Rm-bench: Benchmarking reward models of language models with subtlety and style. *arXiv preprint arXiv:2410.16184*.
- Zihan Liu, Wei Ping, Rajarshi Roy, Peng Xu, Chankyu Lee, Mohammad Shoeybi, and Bryan Catanzaro. 2024c. ChatQA: Surpassing gpt-4 on conversational QA and RAG. In *NeurIPS*.
- I Loshchilov. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.
- Jianqiao Lu, Zhiyang Dou, Hongru Wang, Zeyu Cao, Jianbo Dai, Yingjia Wan, Yinya Huang, and Zhijiang Guo. 2024. Autocv: Empowering reasoning with automated process labeling via confidence variation. *arXiv preprint arXiv:2405.16802*.
- Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Harsh Lara, Yunxuan Li, Lei Shu, Yun Zhu, Lei Meng, Jiao Sun, et al. 2024. Improve mathematical reasoning in language models by automated process supervision. *arXiv preprint arXiv:2406.06592*.
- Ziyang Luo, Can Xu, Pu Zhao, Qingfeng Sun, Xiubo Geng, Wenxiang Hu, Chongyang Tao, Jing Ma, Qingwei Lin, and Daxin Jiang. 2023. Wizardcoder: Empowering code large language models with evol-instruct. *arXiv preprint arXiv:2306.08568*.
- Mistral. 2024. **Math Σ tral**.
- Arindam Mitra, Hamed Khanpour, Corby Rosset, and Ahmed Awadallah. 2024. Orca-math: Unlocking the potential of slms in grade school math. *arXiv preprint arXiv:2402.14830*.
- 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. *arXiv preprint arXiv:2501.19393*.

- Subhabrata Mukherjee, Arindam Mitra, Ganesh Jawahar, Sahaj Agarwal, Hamid Palangi, and Ahmed Awadallah. 2023. Orca: Progressive learning from complex explanation traces of gpt-4. *arXiv preprint arXiv:2306.02707*.
- OpenAI. 2024a. GPT-4o mini: advancing cost-efficient intelligence.
- OpenAI. 2024b. [Hello GPT-4o](#).
- Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.
- Baolin Peng, Chunyuan Li, Pengcheng He, Michel Galley, and Jianfeng Gao. 2023. Instruction tuning with gpt-4. *arXiv preprint arXiv:2304.03277*.
- Amrith Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal, Alekh Agarwal, Jonathan Berant, and Aviral Kumar. 2024. Rewarding progress: Scaling automated process verifiers for llm reasoning. *arXiv preprint arXiv:2410.08146*.
- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, YK Li, Yu Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.
- Jia Tracy Shen, Michiharu Yamashita, Ethan Prihar, Neil Heffernan, Xintao Wu, Ben Graff, and Dongwon Lee. 2021. Mathbert: A pre-trained language model for general nlp tasks in mathematics education. In *NeurIPS 2021 Math AI for Education Workshop*.
- Skywork-o1. 2024. [Skywork-o1 open series](https://huggingface.co/Skywork). <https://huggingface.co/Skywork>.
- Zhengyang Tang, Xingxing Zhang, Benyou Wang, and Furu Wei. 2024. Mathscla: Scaling instruction tuning for mathematical reasoning. In *Forty-first International Conference on Machine Learning*.
- Terence Tao. 2023. [Embracing change and resetting expectations](#).
- Kimi Team, Angang Du, Bofei Gao, Bawei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, et al. 2025. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*.
- Teknium. 2023. [GPTeacher-General-Instruct](#).
- The-Vicuna-Team. 2023. [ShareGPT-Vicuna](#).
- Shubham Toshniwal, Wei Du, Ivan Moshkov, Branislav Kisacanin, Alexan Ayrapetyan, and Igor Gitman. 2024. Openmathinstruct-2: Accelerating ai for math with massive open-source instruction data. *arXiv preprint arXiv:2410.01560*.
- Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shrutu Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.
- Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. 2022. Solving math word problems with process-and outcome-based feedback. *arXiv preprint arXiv:2211.14275*.
- Haoxiang Wang, Wei Xiong, Tengyang Xie, Han Zhao, and Tong Zhang. 2024a. Interpretable preferences via multi-objective reward modeling and mixture-of-experts. In *EMNLP*.
- Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. 2024b. Math-shepherd: Verify and reinforce LLMs step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Bangkok, Thailand. Association for Computational Linguistics.
- Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyang Jiang, et al. 2024c. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *arXiv preprint arXiv:2406.01574*.
- Zengzhi Wang, Rui Xia, and Pengfei Liu. 2023. Generative ai for math: Part i—mathpile: A billion-token-scale pretraining corpus for math. *arXiv preprint arXiv:2312.17120*.
- Zhilin Wang, Alexander Bukharin, Olivier Delalleau, Daniel Egert, Gerald Shen, Jiaqi Zeng, Oleksii Kuchaiev, and Yi Dong. 2024d. [Helpsteer2-preference: Complementing ratings with preferences](#). *Preprint*, arXiv:2410.01257.
- Zhilin Wang, Yi Dong, Olivier Delalleau, Jiaqi Zeng, Gerald Shen, Daniel Egert, Jimmy J. Zhang, Makesh Narsimhan Sreedhar, and Oleksii Kuchaiev. 2024e. [Helpsteer2: Open-source dataset for training top-performing reward models](#). *Preprint*, arXiv:2406.08673.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.
- Yuxiang Wei, Zhe Wang, Jiawei Liu, Yifeng Ding, and Lingming Zhang. 2024. Magicoder: Empowering code generation with oss-instruct. In *Forty-first International Conference on Machine Learning*.
- Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, Qingwei

- Lin, and Daxin Jiang. 2024. [WizardLM: Empowering large pre-trained language models to follow complex instructions](#). In *The Twelfth International Conference on Learning Representations*.
- An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. 2024a. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.
- An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, et al. 2024b. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*.
- Huaiyuan Ying, Shuo Zhang, Linyang Li, Zhejian Zhou, Yunfan Shao, Zhaoye Fei, Yichuan Ma, Jiawei Hong, Kuikun Liu, Ziyi Wang, et al. 2024. InternLM-Math: Open math large language models toward verifiable reasoning. *arXiv preprint arXiv:2402.06332*.
- Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2023. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*.
- Yue Yu, Zhengxing Chen, Aston Zhang, Liang Tan, Chenguang Zhu, Richard Yuanzhe Pang, Yundi Qian, Xuewei Wang, Suchin Gururangan, Chao Zhang, et al. 2024. Self-generated critiques boost reward modeling for language models. *arXiv preprint arXiv:2411.16646*.
- Lifan Yuan, Ganqu Cui, Hanbin Wang, Ning Ding, Xingyao Wang, Jia Deng, Boji Shan, Huimin Chen, Ruobing Xie, Yankai Lin, et al. 2024. Advancing llm reasoning generalists with preference trees. *arXiv preprint arXiv:2404.02078*.
- Xiang Yue, Xingwei Qu, Ge Zhang, Yao Fu, Wenhao Huang, Huan Sun, Yu Su, and Wenhui Chen. 2024a. Mammoth: Building math generalist models through hybrid instruction tuning. *ICLR*.
- Xiang Yue, Tuney Zheng, Ge Zhang, and Wenhui Chen. 2024b. Mammoth2: Scaling instructions from the web. *NeurIPS*.
- Fan Zhang, Chenglu Li, Owen Henkel, Wanli Xing, Sami Baral, Neil Heffernan, and Hai Li. 2024a. Mathllms: Ai cyberinfrastructure with pre-trained transformers for math education. *International Journal of Artificial Intelligence in Education*, pages 1–24.
- Hanning Zhang, Pengcheng Wang, Shizhe Diao, Yong Lin, Rui Pan, Hanze Dong, Dylan Zhang, Pavlo Molchanov, and Tong Zhang. 2024b. Entropy-regularized process reward model.
- Lunjun Zhang, Arian Hosseini, Hritik Bansal, Mehran Kazemi, Aviral Kumar, and Rishabh Agarwal. 2024c. Generative verifiers: Reward modeling as next-token prediction. *arXiv preprint arXiv:2408.15240*.
- Chujie Zheng, Zhenru Zhang, Beichen Zhang, Runji Lin, Keming Lu, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. 2024. Processbench: Identifying process errors in mathematical reasoning. *arXiv preprint arXiv:2412.06559*.
- Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36:46595–46623.

A Additional Analysis of AceInstruct and AceMath-Instruct

A.1 Effectiveness of the Two-Stage training for AceInstruct

As shown in Table 5, we study the effectiveness of two-stage training strategy. For comparison, we use two base models from distinct families (Qwen2.5 and Llama3.1) and conduct single-stage training using either all general SFT data or only the stage-2 SFT data.

We observe that our two-stage training (AceInstruct) consistently outperforms single-stage training. Interestingly, we find notable improvements (more than 3% average score) on a relatively weaker base model (e.g., Llama3.1-8B) compared to a stronger one (e.g., Qwen2.5-7B). This highlights the importance of incorporating extensive coding and math data during training to enhance the model’s ability to handle complex coding and math tasks. We conjecture that the Qwen2.5 models already leverage substantial math and coding SFT data during pretraining, which reduces the effectiveness of an additional stage-1 SFT focused on these areas.

A.2 AceMath-Instruct Using Different Backbone Models

Moreover, in Figure 3, we observe that except for Qwen2.5-1.5B, all the models from “Ours (Base)” outperform the corresponding math-instruct models that use stronger math base models as backbones. This further indicates that smaller models (e.g., 1.5B) rely more on continual pre-training with a large math corpus to enhance their math problem-solving capability.

Table 6 shows the full results of AceMath-Instruct using various models as backbone models. Additionally, we include the results for Llama3.1-8B-Base as the backbone model and compare our model to OpenMath2-Llama3.1-8B (Toshniwal et al., 2024) that also uses Llama3.1-8B-Base as its backbone model. We find that except for our 1.5B model based on Qwen2.5-1.5B-Base, all our models, including those built on base models, outperform their respective strong baselines, often by a significant margin.

A.3 Ablation Studies on Training Data and Strategies for AceMath-Instruct

In Table 7, we conduct ablation studies on training data and strategies across various backbone models

for training our AceMath-Instruct models.

First, we explore the effectiveness of using either GPT-4o-mini responses or Qwen2.5-Math-72B-Instruct responses individually. Given that our best-performing models leverage responses from both, we analyze the impact of relying solely on one model when constructing general and math SFT data. Notably, even when only GPT-4o-mini responses are available, we achieve strong performance, with just a 1% average score drop when Qwen2.5-7B-Base serves as the backbone model. Furthermore, with Llama3.1-8B-Base as the backbone, using responses from GPT-4o-mini, Qwen2.5-Math-72B-Instruct, or their combination (AceMath-Instruct) yields comparable results. This indicates that the robustness of our data construction process which minimizes dependence on super powerful math expert models for generating synthetic data.

Second, we analyze the effectiveness of different math-specific samples for math SFT. To study this, we compare AceMath-Instruct trained with 1.6 million math SFT samples (details in §3.3.2) to models trained using all available math SFT samples (2.3 million) or only cross-checked high-quality samples (800K). We find that simply increasing the quantity of data or exclusively using high-quality samples does not yield better outcomes. Instead, combining cross-checked high-quality data with additional samples that include a diverse range of math questions produces superior results.

Third, we study the impact of conducting general SFT before transitioning to math SFT. To explore this, we skip the general SFT step, and conduct math SFT directly using all math-specific samples. We observe that skipping general SFT typically results in an average score drop of approximately 1%, even when using a math-base model (e.g., Qwen2.5-Math-72B-Base) as the backbone. The results highlight the effectiveness of conducting general SFT prior to math SFT.

A.4 AIME 2024 & AMC 2023 Results

We further evaluate our models on AMC 2023² and AIME 2024³. Although these benchmarks are highly challenging math competition benchmarks, they are quite limited in size, with AMC 2023 containing only 40 test samples and AIME 2024 com-

²<https://huggingface.co/datasets/AI-MO/aimo-validation-amc>

³<https://huggingface.co/datasets/AI-MO/aimo-validation-aime>

Models	HumanEval	MBPP	GSM8K	MATH	MMLU	MMLU Pro	Avg.
AceInstruct-8B-Llama3.1	81.10	74.71	90.45	64.42	68.31	43.27	70.38
▷ Single-Stage SFT w/ all general SFT data	78.66	69.26	87.79	56.80	67.62	42.64	67.13
▷ Single-Stage SFT w/ only stage-2 data	73.78	67.32	88.17	55.84	67.48	42.85	65.91
AceInstruct-7B-Qwen2.5	85.37	74.32	93.10	76.40	74.68	54.50	76.40
▷ Single-Stage SFT w/ all general SFT data	83.54	75.49	91.96	75.04	73.96	53.36	75.56
▷ Single-Stage SFT w/ only stage-2 data	83.54	73.15	92.27	75.12	74.26	53.06	75.23

Table 5: Ablation studies of our AceInstruct general SFT models regarding the effectiveness of the two-stage training strategy.

Models	GSM8K	MATH	Minerva Math	GaoKao 2023 En	Olympiad Bench	College Math	MMLU STEM	Avg.
DeepSeek-Math-7B-RL	88.20	52.40	20.60	43.60	19.00	37.50	64.80	46.59
AceMath-Instruct (backbone: DeepSeek-Coder-7B-Base)	83.85	59.72	29.78	53.51	24.59	44.64	55.95	50.29
AceMath-Instruct (backbone: DeepSeek-Math-7B-Base)	85.06	66.86	40.07	56.62	29.63	48.94	65.53	56.10
Llama-3.1-8B-Instruct	84.50	51.90	21.70	38.40	15.40	33.80	60.50	43.74
OpenMath2-Llama3.1-8B	91.70	67.80	16.91	53.76	28.00	46.13	46.02	50.08
AceMath-Instruct (backbone: Llama3.1-8B-Base)	91.51	69.06	31.99	59.74	32.00	49.08	67.94	57.33
Qwen2.5-Math-1.5B-Instruct	84.80	75.80	29.40	65.50	38.10	47.70	57.50	56.97
AceMath-Instruct (backbone: Qwen2.5-1.5B-Base)	80.89	64.59	30.51	53.25	27.11	47.80	58.62	51.82
AceMath-Instruct (backbone: Qwen2.5-Math-1.5B-Base)	86.95	76.84	41.54	64.42	33.78	54.36	62.04	59.99
Qwen2.5-Math-7B-Instruct	95.20	83.60	37.10	66.80	41.60	46.80	71.90	63.29
AceMath-Instruct (backbone: Qwen2.5-7B-Base)	93.56	77.10	43.38	65.19	37.78	54.90	77.41	64.19
AceMath-Instruct (backbone: Qwen2.5-Math-7B-Base)	93.71	83.14	51.11	68.05	42.22	56.64	75.32	67.17
Qwen2.5-Math-72B-Instruct	95.90	85.90	44.10	71.90	49.00	49.50	80.80	68.16
AceMath-Instruct (backbone: Qwen2.5-72B-Base)	95.99	85.06	54.04	73.25	46.96	57.10	85.48	71.13
AceMath-Instruct (backbone: Qwen2.5-Math-72B-Base)	96.44	86.10	56.99	72.21	48.44	57.24	85.44	71.84

Table 6: Greedy decoding results of AceMath-Instruct across different backbone models.

prising just 30. Following Yang et al. (2024b), we evaluate these benchmarks separately and present the results as follows.

Table 8 shows the greedy decoding results on AIME 2024 and AMC 2023. We find that the AceMath-1.5B/7B-Instruct models slightly outperform Qwen2.5-Math-1.5B/7B-Instruct on both datasets, while AceMath-72B-Instruct falls short of Qwen2.5-Math-72B-Instruct’s performance on AIME 2024. Given that AIME 2024 contains challenging math problems comparable to pre-Olympiad levels, these results indicate that there is room to improve AceMath-Instruct to better address varying levels of mathematical difficulty.

B Additional Analysis of AceMath-RM

B.1 Ablation Studies on AceMath-RM

In Table 9, we conduct ablation studies on the model backbone, data sampling method, and different loss functions used to train the reward model. First, we found that using AceMath-7B-Instruct as the backbone model for training the model consistently outperforms Qwen2.5-Math-7B-Instruct on average of 7 datasets, with a similar performance gap observed at the 72B scale. Secondly, we ob-

served that employing reward score-sorted sampling (§4.1.2) during the data construction process improves average accuracy compared to random sampling. This highlights the benefits of filtering out noisy labels when heuristic evaluation toolkits produce false negative errors. Lastly, we experimented with different loss functions. We found that using pairwise Bradley-Terry loss achieves comparable accuracy to the listwise loss approach, however requiring $3.7\times$ more training time using 8 H100 GPUs. Additionally, training a classifier using cross-entropy loss or a regression model using mean squared error (MSE) loss both resulted in lower accuracy. A similar performance gap was also observed at the 72B scale for the cross-entropy classification approach. Since the data is constructed for the listwise BT approach, where each problem consists of six responses, this also leads to 3.8 times more compute hours on 8 GPUs.

B.2 AceMath-RM Results on $rm@k$

In Figure 4, we present a comparison between AceMath-72B-RM and Qwen2.5-Math-RM-72B on $rm@k$ ($k = 8, 16, 32, 64, 128$) across the seven datasets listed in Table 3, using samples generated

Models	GSM8K	MATH	Minerva Math	GaoKao 2023 En	Olympiad Bench	College Math	MMLU STEM	Avg.
Backbone: Llama3.1-8B-Base								
AceMath-Instruct	91.51	69.06	31.99	59.74	32.00	49.08	67.94	57.33
▷ Only Qwen2.5-Math-72B-Instruct	91.13	69.66	33.82	60.26	30.37	49.86	66.21	57.33
▷ Only GPT-4o-mini	90.83	68.12	36.03	60.26	31.70	48.05	66.50	57.36
▷ Skipping general SFT	91.81	68.70	31.99	59.48	31.11	48.40	62.76	56.32
Backbone: Qwen2.5-7B-Base								
AceMath-Instruct	93.56	77.10	43.38	65.19	37.78	54.90	77.41	64.19
▷ Only Qwen2.5-Math-72B-Instruct	92.80	76.96	41.91	63.64	38.07	54.93	75.64	63.42
▷ Only GPT-4o-mini	91.66	74.14	43.75	64.42	39.26	52.27	76.03	63.08
▷ Math SFT with all math samples	93.40	77.12	42.28	65.19	37.78	54.05	75.33	63.59
▷ Math SFT with only cross-checked samples	92.72	76.76	41.54	65.97	36.74	54.33	76.78	63.55
▷ Skipping general SFT	93.03	77.52	40.44	62.86	37.19	54.58	75.77	63.06
Backbone: Qwen2.5-Math-72B-Base								
AceMath-Instruct	96.44	86.10	56.99	72.21	48.44	57.24	85.44	71.84
▷ Math SFT with all math samples	96.29	86.06	55.15	70.13	46.67	57.49	84.96	70.96
▷ Skipping general SFT	95.75	85.52	56.25	71.43	45.33	56.71	84.42	70.77

Table 7: Ablation Studies on training data and strategies across various backbone models for training our AceMath-Instruct models. The ablation studies can be categorized into three parts: 1) evaluating the effectiveness of using either GPT-4o-mini responses or Qwen2.5-Math-72B-Instruct responses individually; 2) analyzing the effectiveness of different math-specific samples for math SFT; and 3) assessing the impact of conducting general SFT prior to math SFT.

Models	AIME 2024	AMC 2023
Llama-3.1-405B-Instruct	5/30	20/40
Claude 3.5 Sonnet (2024-1022)	4/30	21/40
OpenMath2-Llama3.1-8B	3/30	16/40
OpenMath2-Llama3.1-70B	4/30	20/40
Qwen2.5-Math-1.5B-Instruct	3/30	24/40
Qwen2.5-Math-7B-Instruct	5/30	25/40
Qwen2.5-Math-72B-Instruct	9/30	28/40
AceMath-1.5B-Instruct	4/30	25/40
AceMath-7B-Instruct	6/30	26/40
AceMath-72B-Instruct	6/30	28/40

Table 8: Greedy decoding results of AceMath-Instruct on AIME 2024 and AMC 2023.

by AceMath-7B-Instruct. We report the average accuracy across these seven datasets, each with 10 different random seeds.

First, we find that using AceMath-72B-RM to score the outputs from AceMath-7B-Instruct consistently improves the average accuracy, increasing from 72.6 to 74.4 as k rises from 8 to 128.

Second, we observe that AceMath-RM consistently outperforms Qwen2.5-Math-RM in scoring outputs generated from AceMath-7B-Instruct, and this improvement becomes larger as k increases.

Furthermore, we compare the performance of AceMath-72B-RM paired with AceMath-Instruct to Qwen2.5-Math-RM-72B paired with Qwen2.5-Math-Instruct. As shown in Figure 1, the AceMath combination consistently outperforms its Qwen2.5 counterpart in terms of $rm@8$, on average, for both the 7B and 72B models. Remarkably, we find that the our AceMath-7B model even outperforms the

Model	AceMath-RewardBench
AceMath-7B-RM	67.41
▷ Backbone: Qwen2.5-Math-7B-Instruct	66.93
▷ Data: Random sampling	67.07
▷ Loss: Pairwise BT	67.33
▷ Loss: Cross-entropy Classification	66.93
▷ Loss: MSE Regression	66.79

AceMath-72B-RM	69.53
▷ Backbone: Qwen2.5-Math-72B-Instruct	69.09
▷ Loss: Cross-entropy Classification	68.66

Table 9: Ablation study of AceMath-7/72B-RM on AceMath-RewardBench (Backbone: AceMath-7/72B-Instruct; Data: reward score-sorted sampling; Loss: list-wise Bradley-Terry).

Qwen2.5-Math-72B in $rm@8$, showing the potential of a smaller model when paired with a carefully designed reward model.

B.3 Learning curves of reward model training for AceMath-RM

In Figure 5, we aim to understand how reward modeling accuracy improves as we increase model size and use additional data. We find distinct patterns in the interplay between model size and data scaling. In general on simpler dataset such as GSM8K, all model sizes (ranging from 0.5B to 32B parameters) exhibit steady improvements as training proceeds, with larger models achieving higher accuracy. In contrast, on the more challenging datasets, which requires college-level knowledge, such as Minerva Math, MMLU STEM, and OlympiadBench, model size emerges as a critical factor: smaller models (*e.g.*, 0.5B, 1.5B) show negligible improvement de-

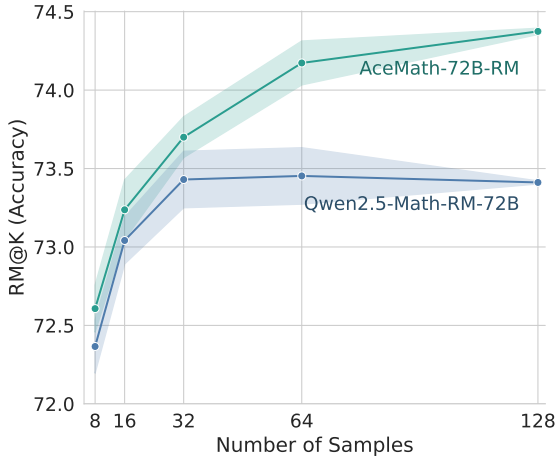


Figure 4: rm@k evaluation on average accuracy of 7 datasets for AceMath-7B-Instruct.

spite increased data, whereas larger models (*e.g.*, 14B, 32B) achieve better accuracy gains. These results suggest that increasing model size provides the greatest benefit, whereas the advantages of increasing data appear less pronounced. Our experiments use Qwen2.5-Instruct (Yang et al., 2024a) model family. instead of Qwen2.5-Math-Instruct, as it provides a more comprehensive set of models with different sizes. All models are trained for one epoch only.

C Related Work

C.1 Continued Pre-training on Math Corpus

Many studies have investigated the integration of large-scale mathematical data for pre-training LLMs to enhance their math capabilities (Shen et al., 2021; Wang et al., 2023; Zhang et al., 2024a; Ying et al., 2024; Akter et al., 2024; Hui et al., 2024). Additionally, some research has focused on developing math-specialized LLMs by continuing the pre-training of a general-purpose LLM with an extensive math corpus, sourced from math-related web texts, encyclopedias, exam questions, and synthetic mathematical data (Shao et al., 2024; Yang et al., 2024b). These works demonstrate that this additional math-focused pre-training significantly enhances the model’s ability to solve math problems, benefiting not only the pre-trained *base* model but also subsequent *instruct* models after post-training.

C.2 Supervised Fine-Tuning

Numerous supervised fine-tuning (SFT) datasets have been developed to enhance pretrained LLMs

with versatile capability, such as instruction following (Chiang et al., 2023; The-Vicuna-Team, 2023; Lian et al., 2023; Mukherjee et al., 2023; Teknium, 2023; Peng et al., 2023; Yuan et al., 2024), coding (Glaive-AI, 2023; Wei et al., 2024; Luo et al., 2023), and mathematical problem-solving (Yue et al., 2024a,b; Yu et al., 2023; Mitra et al., 2024; Li et al., 2024b). Due to the high cost of human-annotated data, synthetic data generation has become an essential component of SFT data construction, including both prompt and response augmentation (Yu et al., 2023; Xu et al., 2024; Luo et al., 2023; Li et al., 2024a; Toshniwal et al., 2024).

Taking this further, math-instructed models have been developed to advance LLM performance in the mathematics domain (Shao et al., 2024; Toshniwal et al., 2024; Yang et al., 2024b) by utilizing math-specific pretrained models as backbones and vast amounts of synthetic post-training data tailored to mathematics. For example, OpenMath-Instruct (Toshniwal et al., 2024) shows that math-specialized SFT with extensive synthetic data on the Llama3.1 base model significantly outperforms the corresponding Llama3.1 instruct model on mathematical benchmarks. In addition, Qwen2.5-Math (Yang et al., 2024b) demonstrates that a 7B math-instruct model can achieve math reasoning capabilities comparable to GPT-4o. Reinforcement learning (RL) has recently emerged as a pivotal technique in enhancing the math reasoning performance, which allows models to iteratively refine their outputs based on the correctness of generated solutions (Team et al., 2025; Guo et al., 2025).

C.3 Reward Modeling

Training reward models for mathematical verification often involves discriminative approaches, such as binary classification to distinguish correct solutions from incorrect ones (Cobbe et al., 2021). Alternatively, preference-based methods are employed, leveraging techniques like the Bradley-Terry loss (Bradley and Terry, 1952; Ouyang et al., 2022) or regression loss to rank solutions, as demonstrated in models like HelpSteer (Wang et al., 2024e,d). In contrast, generative reward models, such as LLM-as-a-judge (Zheng et al., 2023) prompt LLMs to act as verifiers using predefined rubrics and grading templates (Bai et al., 2022), GenRM (Zhang et al., 2024c) leverages Chain-of-Thought reasoning (Wei et al., 2022), and CriticRM (Yu et al., 2024) uses critic before predicting a reward. Our work on outcome reward model

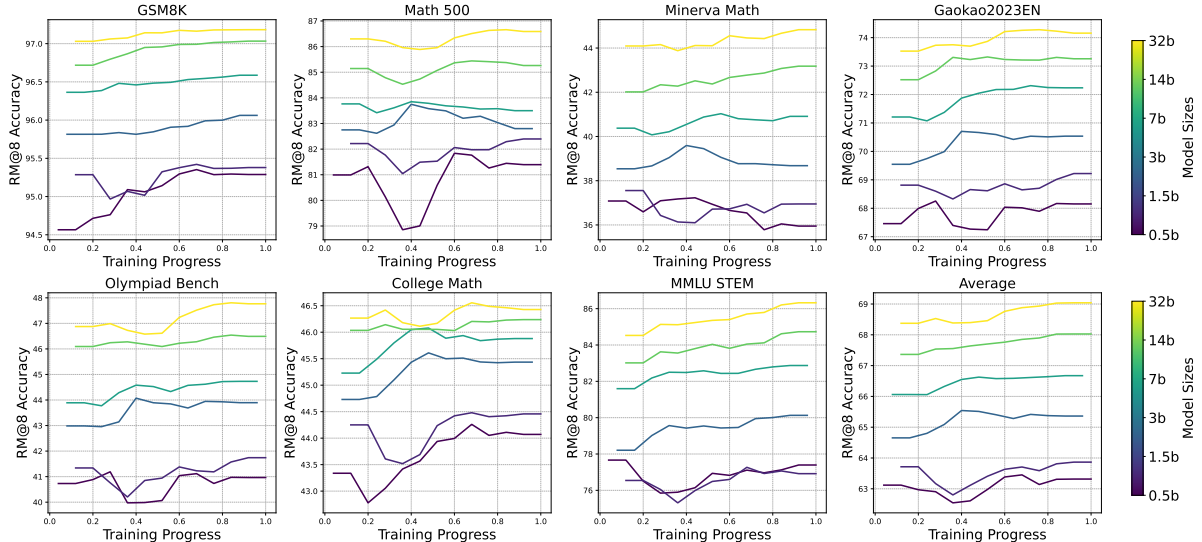


Figure 5: Learning curves for reward model training. All models are trained from Qwen2.5-Instruct family.

mainly focuses on robustness against style biases (Liu et al., 2024b) by sampling diverse model responses for training. Beyond outcome-based reward models, process reward models (PRMs) provide step-by-step evaluations of model responses (Uesato et al., 2022; Lightman et al., 2023). For example, Math-Shepherd (Wang et al., 2024b) introduces an automated sampling method to construct large-scale process supervision data for training, following by further developments in step-wise supervision labeling (Dong et al., 2024), including PAV (Setlur et al., 2024), OmegaPRM (Luo et al., 2024), ER-PRM (Zhang et al., 2024b), AutoPSV (Lu et al., 2024) and ProcessBench (Zheng et al., 2024).

D Details of General and Math SFT Data and Training Strategies

D.1 Synthetic Prompt Generation for Math SFT

We generate additional synthetic prompts to enrich the diversity of our math prompt collection. This process involves two key steps: 1) leveraging diverse seed prompts to inspire a powerful *instruct* model to generate entirely new, potentially more challenging or uncommon prompts, and 2) ensuring that the generated prompts are solvable, as unsolvable prompts can lead to incorrect answers, which may degrade performance when used for training. Therefore, we select NuminaMath as our seed prompt source due to its broad coverage of math questions across various difficulty levels. Then, we apply two strategies inspired by Xu et al. (2024): *in-breadth evolution* for generating more

rare prompts and *in-depth evolution* for generating more challenging ones. For synthetic prompt generation, we utilize GPT-4o-mini (2024-0718).

It is crucial to filter out low-quality synthetic prompts. In particular, we find that one type of *in-depth evolution*, which involves adding constraints to existing prompts to generate new ones, can sometimes produce unsolvable or overly challenging questions. This, in turn, may result in incorrect answers being included in the training data, ultimately degrading model performance (see ablation studies in §3.6.3). As a result, we exclude this type of prompt augmentation. Moreover, we filter out the synthetic prompts exceeding 300 words, as excessively lengthy math-related prompts are often problematic or unsolvable. Finally, we refine the synthetic math prompts to approximately one million by filtering out 500K, ensuring a more curated dataset for training. As a result, we have a total collection of over 2.3 million math prompts (1.3M initial prompts + 1M synthetic prompts).

In the next parts, we describe the prompt we provided to GPT-4o-mini (2024-0718) for generating synthetic prompts tailored to math SFT. We utilize the in-breath evolution and in-depth evolution prompts inspired from Xu et al. (2024).

D.1.1 In-Breath Evolution

We use the following prompt to generate more diverse math questions.

You are a good math question creator.

Your objective is to draw inspiration from the #Given MATH Question# to create

a brand new math question. This new math question should be distinctly different from the #Given MATH Question# and be even more unique.

The length and difficulty level of the #Created MATH Question# should be similar to those of the #Given MATH Question#.

The #Created MATH Question# must be solvable and understandable by humans.

#Given MATH Question#:
{given_math_question}

#Created MATH Question#:

D.1.2 In-Depth Evolution

We use the following prompt to generate more challenging math questions.

You are a good math question creator.

Your objective is to draw inspiration from the #Given MATH Question# to create a brand new math question. This new math question should be more complex and challenging than the #Given MATH Question#.

The #Created MATH Question# must be solvable and understandable by humans.

#Given MATH Question#:
{given_math_question}

#Created MATH Question#:

Moreover, we find that the following prompt that requires to add constraints to the given prompt could result in unsolvable or overly challenging math questions. This, in turn, can lead to incorrect answers being included in the training data, ultimately degrading model performance.

You are a good math question creator.

Your objective is to rewrite the #Given MATH Question# into a brand new but more complex version. You can complicate the #Given MATH Question# by introducing additional constraints and requirements.

The #Created MATH Question# must be solvable and understandable by humans.

#Given MATH Question#:
{given_math_question}

#Created MATH Question#:

D.2 Response Construction for Math Prompts

We utilize Qwen2.5-Math-72B-Instruct for generating responses to math prompts, given its remarkable performance across various math benchmarks. We add the instruction, “*Please reason step by step, and put your final answer within `\boxed{\}`.*” to the prompt to ensure the responses are presented in a clear, step-by-step format with a consistent style.

We generate a single response for each of the over 2.3M prompts and ensure consistency in the response format by selecting only those responses (along with their prompts) that adhere to a uniform structure (e.g., starting the response with a summary of the question and having the final answer within `\boxed{\}`). Additionally, responses exceeding 2,500 words are excluded, along with their prompts, as excessive response length often indicates a verbose or incorrect solution, or an unfinished response. Furthermore, while Qwen2.5-Math-72B-Instruct demonstrates strong capabilities, it occasionally produces repetitive strings (e.g., repeating the same text until reaching the maximum output length). We detect and remove such patterns, along with their corresponding prompts. Although these cases represent only a small fraction of the dataset, they can negatively impact the final performance and are carefully filtered during the curation process. After filtering, we obtain a total of around 2.3 million math SFT samples (1.83 billion tokens), of which around 1.2 million are utilized in the general SFT.

Qwen2.5-Math-72B-Instruct can still generate incorrect solutions which may negatively impact model training. To mitigate this, we focus on identifying samples with accurate final answers to create a higher-quality dataset for training.

Our approach involves cross-checking answers generated by different models and treating solutions with consistent outcomes as highly likely to be correct. Specifically, we leverage another strong model, GPT-4o-mini (2024-0718), to generate responses. Since GPT-4o-mini is comparatively weaker in mathematics than Qwen2.5-Math-72B-

Instruct, we generate two responses per prompt and consider answers consistent across both responses as potentially correct. Finally, we compare these answers with those from Qwen2.5-Math-72B-Instruct, and select matched final answers as high-quality solutions for training, which results in a total size of 800K math SFT samples.

D.3 Data Decontamination

Data decontamination is essential in SFT to ensure unbiased evaluation and to prevent models from memorizing test samples. Following Yang et al. (2024b), we conduct data decontamination for math SFT prompts. The process begins with text normalization and the removal of irrelevant punctuation for each math prompt. Next, we filter out the prompt that has a 13-gram overlap with the test data and the longest common subsequence exceeding 60% of its length. For the rest of non-math SFT prompts, we simply filter out those with a 13-gram overlap with test samples.

D.4 General SFT Training Strategy

We organize the constructed general SFT data (around three million samples) to support this two-stage training. For stage-1, the majority of the coding and math samples are selected, leading to a total of around 2 million SFT samples. Stage-2 training utilizes the remaining coding and math SFT samples, a subset of the stage-1 data, along with all other general SFT samples, resulting in a total of around 1.6 million samples. For math SFT samples used in stage-2 training, we select only the cross-checked high-quality data where the final answers provided by GPT-4o-mini and Qwen2.5-Math-72B-Instruct align, as detailed in §3.2.2. This strategy ensures that stage-2 training integrates additional, diverse, and high-quality coding and math SFT samples, thereby fostering a more robust model.

D.5 SFT Training Hyperparameters

All SFT models are trained using the AdamW optimizer (Kingma, 2014; Loshchilov, 2017). We use a learning rate of $5e-6$ for the general SFT and $3e-6$ for the math SFT. A global batch size of 128 is used across all model sizes, except for the 72B model, where it is increased to 256. We conduct one epoch of training with a maximum sequence length of 4096 for both general SFT and math SFT.

E Chain-of-Thought Reasoning Examples

Below, we present several Chain-of-Thought reasoning examples generated by AceMath-72B-Instruct.

SAMPLE 1

Question:

For each positive integer N , let $P(N)$ denote the product of the digits of N . For example, $P(8) = 8$, $P(451) = 20$, and $P(2023) = 0$. Compute the least positive integer n such that $P(n + 23) = P(n) + 23$.

AceMath-72B-Instruct:

To find the least positive integer n such that $P(n + 23) = P(n) + 23$, we need to analyze the properties of the function P , which gives the product of the digits of a number. Let's start by considering the possible values of n and their corresponding values of $P(n)$ and $P(n + 23)$.

First, note that if n has a digit of 0, then $P(n) = 0$. In this case, $P(n + 23) = 23$, which means $n + 23$ must have digits whose product is 23. Since 23 is a prime number, the only way to achieve this is if $n + 23$ is either 23 or a number with 2 and 3 as its only digits. However, n must be a positive integer, so $n + 23$ cannot be 23. Therefore, $n + 23$ must be a number like 230, 203, 320, 302, etc. The smallest such number is 230, so $n = 230 - 23 = 207$. Let's verify this solution:

$$- P(207) = 2 \cdot 0 \cdot 7 = 0$$

$$- P(230) = 2 \cdot 3 \cdot 0 = 0$$

This does not satisfy the equation $P(n + 23) = P(n) + 23$. Therefore, we need to consider other possibilities. Next, let's consider the case where n does not have a digit of 0. We need to find n such that the product of the digits of $n + 23$ is 23 more than the product of the digits of n . Let's try some small values of n :

$$- \text{If } n = 1, \text{ then } P(1) = 1 \text{ and } P(24) = 2 \cdot 4 = 8. \\ 8 \neq 1 + 23.$$

$$- \text{If } n = 2, \text{ then } P(2) = 2 \text{ and } P(25) = 2 \cdot 5 = 10. \\ 10 \neq 2 + 23.$$

$$- \text{If } n = 3, \text{ then } P(3) = 3 \text{ and } P(26) = 2 \cdot 6 = 12. \\ 12 \neq 3 + 23.$$

$$- \text{If } n = 4, \text{ then } P(4) = 4 \text{ and } P(27) = 2 \cdot 7 = 14. \\ 14 \neq 4 + 23.$$

$$- \text{If } n = 5, \text{ then } P(5) = 5 \text{ and } P(28) = 2 \cdot 8 = 16. \\ 16 \neq 5 + 23.$$

$$- \text{If } n = 6, \text{ then } P(6) = 6 \text{ and } P(29) = 2 \cdot 9 =$$

18. $18 \neq 6 + 23$.

- If $n = 7$, then $P(7) = 7$ and $P(30) = 3 \cdot 0 = 0$.
 $0 \neq 7 + 23$.

- If $n = 8$, then $P(8) = 8$ and $P(31) = 3 \cdot 1 = 3$.
 $3 \neq 8 + 23$.

- If $n = 9$, then $P(9) = 9$ and $P(32) = 3 \cdot 2 = 6$.
 $6 \neq 9 + 23$.

- If $n = 10$, then $P(10) = 1 \cdot 0 = 0$ and
 $P(33) = 3 \cdot 3 = 9$. $9 \neq 0 + 23$.

- If $n = 11$, then $P(11) = 1 \cdot 1 = 1$ and
 $P(34) = 3 \cdot 4 = 12$. $12 \neq 1 + 23$.

- If $n = 12$, then $P(12) = 1 \cdot 2 = 2$ and
 $P(35) = 3 \cdot 5 = 15$. $15 \neq 2 + 23$.

- If $n = 13$, then $P(13) = 1 \cdot 3 = 3$ and
 $P(36) = 3 \cdot 6 = 18$. $18 \neq 3 + 23$.

- If $n = 14$, then $P(14) = 1 \cdot 4 = 4$ and
 $P(37) = 3 \cdot 7 = 21$. $21 = 4 + 17$.

- If $n = 15$, then $P(15) = 1 \cdot 5 = 5$ and
 $P(38) = 3 \cdot 8 = 24$. $24 = 5 + 19$.

- If $n = 16$, then $P(16) = 1 \cdot 6 = 6$ and
 $P(39) = 3 \cdot 9 = 27$. $27 = 6 + 21$.

- If $n = 17$, then $P(17) = 1 \cdot 7 = 7$ and
 $P(40) = 4 \cdot 0 = 0$. $0 \neq 7 + 23$.

- If $n = 18$, then $P(18) = 1 \cdot 8 = 8$ and
 $P(41) = 4 \cdot 1 = 4$. $4 \neq 8 + 23$.

- If $n = 19$, then $P(19) = 1 \cdot 9 = 9$ and
 $P(42) = 4 \cdot 2 = 8$. $8 \neq 9 + 23$.

- If $n = 20$, then $P(20) = 2 \cdot 0 = 0$ and
 $P(43) = 4 \cdot 3 = 12$. $12 \neq 0 + 23$.

- If $n = 21$, then $P(21) = 2 \cdot 1 = 2$ and
 $P(44) = 4 \cdot 4 = 16$. $16 \neq 2 + 23$.

- If $n = 22$, then $P(22) = 2 \cdot 2 = 4$ and
 $P(45) = 4 \cdot 5 = 20$. $20 = 4 + 16$.

- If $n = 23$, then $P(23) = 2 \cdot 3 = 6$ and
 $P(46) = 4 \cdot 6 = 24$. $24 = 6 + 18$.

- If $n = 24$, then $P(24) = 2 \cdot 4 = 8$ and
 $P(47) = 4 \cdot 7 = 28$. $28 = 8 + 20$.

- If $n = 25$, then $P(25) = 2 \cdot 5 = 10$ and
 $P(48) = 4 \cdot 8 = 32$. $32 = 10 + 22$.

- If $n = 26$, then $P(26) = 2 \cdot 6 = 12$ and
 $P(49) = 4 \cdot 9 = 36$. $36 = 12 + 24$.

- If $n = 27$, then $P(27) = 2 \cdot 7 = 14$ and
 $P(50) = 5 \cdot 0 = 0$. $0 \neq 14 + 23$.

- If $n = 28$, then $P(28) = 2 \cdot 8 = 16$ and
 $P(51) = 5 \cdot 1 = 5$. $5 \neq 16 + 23$.

- If $n = 29$, then $P(29) = 2 \cdot 9 = 18$ and
 $P(52) = 5 \cdot 2 = 10$. $10 \neq 18 + 23$.

- If $n = 30$, then $P(30) = 3 \cdot 0 = 0$ and
 $P(53) = 5 \cdot 3 = 15$. $15 \neq 0 + 23$.

- If $n = 31$, then $P(31) = 3 \cdot 1 = 3$ and
 $P(54) = 5 \cdot 4 = 20$. $20 = 3 + 17$.

- If $n = 32$, then $P(32) = 3 \cdot 2 = 6$ and
 $P(55) = 5 \cdot 5 = 25$. $25 = 6 + 19$.

- If $n = 33$, then $P(33) = 3 \cdot 3 = 9$ and
 $P(56) = 5 \cdot 6 = 30$. $30 = 9 + 21$.

- If $n = 34$, then $P(34) = 3 \cdot 4 = 12$ and
 $P(57) = 5 \cdot 7 = 35$. $35 = 12 + 23$.

Thus, the least positive integer n such that
 $P(n + 23) = P(n) + 23$ is $\boxed{34}$.

SAMPLE 2

Question:

Consider the function $f(x) = x^2 - 2x$. Determine all real numbers x that satisfy the equation $f(f(f(x))) = 3$.

AceMath-72B-Instruct:

To solve the equation $f(f(f(x))) = 3$ where $f(x) = x^2 - 2x$, we need to determine the values of x that satisfy this equation step by step.

First, let's find the values of x that satisfy $f(x) = 3$. We start with the equation:

$$x^2 - 2x = 3$$

Rearranging terms, we get:

$$x^2 - 2x - 3 = 0$$

This quadratic equation can be factored as:

$$(x - 3)(x + 1) = 0$$

So, the solutions are:

$$x = 3 \quad \text{or} \quad x = -1$$

This means that $f(x) = 3$ if and only if $x = 3$ or $x = -1$.

Next, we need to find the values of x such that $f(f(x)) = 3$. From the previous step, we know that $f(f(x)) = 3$ if and only if $f(x) = 3$ or $f(x) = -1$.

We already know the solutions to $f(x) = 3$, which are $x = 3$ and $x = -1$. Now, let's find the solutions to $f(x) = -1$:

$$x^2 - 2x = -1$$

Rearranging terms, we get:

$$x^2 - 2x + 1 = 0$$

This quadratic equation can be factored as:

$$(x - 1)^2 = 0$$

So, the solution is:

$$x = 1$$

Therefore, the solutions to $f(f(x)) = 3$ are $x = 3$, $x = -1$, and $x = 1$.

Finally, we need to find the values of x such that $f(f(f(x))) = 3$. From the previous step, we know that $f(f(f(x))) = 3$ if and only if $f(f(x)) = 3$ or $f(f(x)) = -1$.

We already know the solutions to $f(f(x)) = 3$, which are $x = 3$, $x = -1$, and $x = 1$. Now, let's find the solutions to $f(f(x)) = -1$:

$$f(f(x)) = -1$$

From the previous step, we know that $f(x) = 3$ or $f(x) = -1$ or $f(x) = 1$ will satisfy $f(f(x)) = 3$. We already know the solutions to $f(x) = 3$, which are $x = 3$ and $x = -1$. We also know the solution to $f(x) = -1$, which is $x = 1$. Now, let's find the solutions to $f(x) = 1$:

$$x^2 - 2x = 1$$

Rearranging terms, we get:

$$x^2 - 2x - 1 = 0$$

This quadratic equation can be solved using the quadratic formula $x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$, where $a = 1$, $b = -2$, and $c = -1$:

$$x = \frac{2 \pm \sqrt{4 + 4}}{2} = \frac{2 \pm \sqrt{8}}{2} = \frac{2 \pm 2\sqrt{2}}{2} = 1 \pm \sqrt{2}$$

So, the solutions are:

$$x = 1 + \sqrt{2} \quad \text{or} \quad x = 1 - \sqrt{2}$$

Therefore, the solutions to $f(f(f(x))) = 3$ are $x = 3$, $x = -1$, $x = 1$, $x = 1 + \sqrt{2}$, and $x = 1 - \sqrt{2}$.

The final answer is:

$$\boxed{3, -1, 1, 1 + \sqrt{2}, 1 - \sqrt{2}}$$

SAMPLE 3

Question:

Alice chooses a set A of positive integers. Then Bob lists all finite nonempty sets B of positive integers with the property that the maximum element of B belongs to A . Bob's list has 2024

sets. Find the sum of the elements of A .

AceMath-72B-Instruct:

To solve the problem, we need to determine the set A such that the number of finite nonempty sets B of positive integers, where the maximum element of B belongs to A , is 2024. Let's denote the elements of A as a_1, a_2, \dots, a_n where $a_1 < a_2 < \dots < a_n$.

For each element a_i in A , the sets B that have a_i as their maximum element are all subsets of $\{1, 2, \dots, a_i\}$ that include a_i . The number of such subsets is 2^{a_i-1} because we can either include or exclude each of the $a_i - 1$ elements $1, 2, \dots, a_i - 1$.

Therefore, the total number of sets B is the sum of 2^{a_i-1} for all i from 1 to n :

$$2^{a_1-1} + 2^{a_2-1} + \dots + 2^{a_n-1} = 2024.$$

We need to find a set A such that the above sum equals 2024. Notice that the sum is a sum of distinct powers of 2, which means it is the binary representation of 2024. Let's convert 2024 to binary:

$$2024_{10} = 11111101000_2.$$

This binary number can be expanded as:

$$2024 = 2^{10} + 2^9 + 2^8 + 2^7 + 2^6 + 2^5 + 2^3.$$

Thus, the corresponding set A is:

$$A = \{11, 10, 9, 8, 7, 6, 4\}.$$

The sum of the elements of A is:

$$11 + 10 + 9 + 8 + 7 + 6 + 4 = 55.$$

Therefore, the sum of the elements of A is $\boxed{55}$.

SAMPLE 4

Question:

How many 'r' in 'strawberry'?

AceMath-72B-Instruct:

To determine how many 'r' letters are in the word "strawberry," we can follow these steps:

1. Write down the word: "strawberry."
2. Identify each 'r' in the word.
3. Count the number of 'r's.

Let's go through the word "strawberry" step by step:

- The first letter is 's'.
- The second letter is 't'.
- The third letter is 'r'.
- The fourth letter is 'a'.
- The fifth letter is 'w'.
- The sixth letter is 'b'.
- The seventh letter is 'e'.
- The eighth letter is 'r'.
- The ninth letter is 'r'.
- The tenth letter is 'y'.

Now, let's count the 'r's:

- There is one 'r' in the third position.
- There is one 'r' in the eighth position.
- There is one 'r' in the ninth position.

So, the total number of 'r's in the word "strawberry" is $1 + 1 + 1 = 3$.

Therefore, the number of 'r' in 'strawberry' is 3.
