Unleashing LLM Reasoning Capability via Scalable Question Synthesis from Scratch

Yuyang Ding[1], **Xinyu Shi**[1], **Xiaobo Liang**[1], **Juntao Li**[1]*
Zhaopeng Tu[2], **Qiaoming Zhu**[1], **Min Zhang**[1]
[1]Soochow University [2]Tencent
{yyding23,xyshi02}@stu.suda.edu.cn
{xbliang,ljt,qmzhu,minzhang}@suda.edu.cn zptu@tencent.com

Abstract

Improving the mathematical reasoning capabilities of Large Language Models (LLMs) is critical for advancing artificial intelligence. However, access to extensive, diverse, and high-quality reasoning datasets remains a significant challenge, particularly for the open-source community. In this paper, we propose ScaleQuest, a novel, scalable, and cost-effective data synthesis method that enables the generation of large-scale mathematical reasoning datasets using lightweight 7B-scale models. ScaleQuest introduces a two-stage question-tuning process comprising Question Fine-Tuning (QFT) and Question Preference Optimization (QPO) to unlock the question generation capabilities of problem-solving models. By generating diverse questions from scratch – without relying on powerful proprietary models or seed data – we produce a dataset of 1 million problem-solution pairs. Our experiments demonstrate that models trained on our data outperform existing open-source datasets in both in-domain and out-of-domain evaluations. Furthermore, our approach shows continued performance improvement as the volume of training data increases, highlighting its potential for ongoing data scaling. The extensive improvements observed in code reasoning tasks demonstrate the generalization capabilities of our proposed method. Our work provides the open-source community with a practical solution to enhance the mathematical reasoning abilities of LLMs.[1]

1 Introduction

How to improve the mathematical reasoning capabilities of Large Language Models (LLMs) has attracted significant attention. The success of recent advanced models, such as OpenAI o1 and Claude-3.5, heavily depends on access to extensive, diverse, and high-quality reasoning datasets. However, the proprietary nature of the data presents a significant barrier to the open-source community. Recent works have highlighted data synthesis as a promising approach (Ntoutsi et al., 2020) to address data scarcity for instruction tuning (Inan et al., 2023). As recent works have disclosed that crafting the right questions is crucial for eliciting the reasoning capabilities of LLMs (Yu et al., 2023a; Shah et al., 2024), the core of reasoning data synthesis lies in creating large-scale and novel questions.

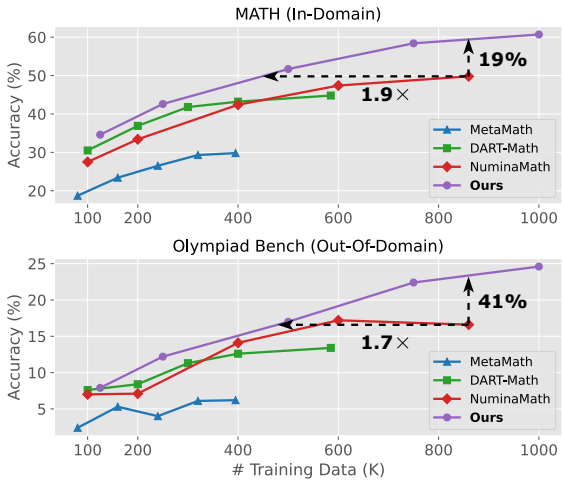

Figure 1: Results of Llama3-8B-Base fine-tuned on publicly available datasets in MATH and OlympiadBench. Our approach demonstrates strong scalability and significant potential for further improvement.

Previous efforts in reasoning data synthesis have demonstrated the effectiveness of leveraging powerful language models to generate instructions. We categorize these approaches into two types: question-driven approaches and knowledge-driven approaches. Question-driven methods include question rephrasing (Yu et al., 2023a), evol-instruct (Xu et al., 2023; Luo et al., 2023; Zeng et al., 2024), question back-translation (Lu et al., 2024), or providing few-shot examples (Mitra et al., 2024). These methods are limited in data diversity, as the generated problems closely resemble the seed ques-

* Corresponding author
[1]Project Page: https://scalequest.github.io.

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 13414–13438

tions, with only minor modifications, such as added conditions or numerical changes. This lack of diversity hampers their potential for scalability. To improve question diversity, recent knowledge-driven works (Huang et al., 2024b) scale question synthesis by constructing knowledge bases (Li et al., 2024b) or concept graphs (Tang et al., 2024) and sampling key points (Huang et al., 2024a) from them to generate new questions. Nevertheless, the above two types of approaches rely on strong models, like GPT-4, to synthesize new questions, but the high API costs make it impractical to generate large-scale data. Despite these advancements, the open-source community still lacks high-quality data at scale and cost-effective synthesis methods.

To meet this requirement, we explore a scalable, low-cost method for data synthesis. We observe that using problem-solving models to directly synthesize reasoning questions, as explored in Yu et al. (2023b) and Xu et al. (2024), falls short in synthesizing reasoning data, as shown in Figure 1 (see Llama3-8B-Magpie results). Accordingly, we propose a novel, scalable, and cost-effective data synthesis method, ScaleQuest, which first introduces a two-stage question-tuning process consisting of Question Fine-Tuning (QFT) and Question Preference Optimization (QPO) to unlock the question generation capability of problem-solving models. Once fine-tuned, these models can then generate diverse questions by sampling from a broad search space without the need for additional seed questions or knowledge constraints. The generated questions can be further refined through a filtering process, focusing on language clarity, solvability, and appropriate difficulty. Moreover, we introduce an extra reward-based filtering strategy to select high-quality responses.

We generate data using lightweight 7B-scale models, producing a final dataset of 1 million question-answer pairs. As shown in Figure 1, compared with other publicly available datasets such as MetaMath (Yu et al., 2023a), DART-Math (Tong et al., 2024), and NuminaMath (Li et al., 2024c), our approach demonstrates great scalability in both in-domain and out-of-domain evaluation. In terms of in-domain evaluation, our method outperforms existing high-quality open-source datasets, achieving better results with the same amount of data. For out-of-domain evaluation, the performance of our approach continues to show promising trends as the volume of training data increases, indicating significant potential for further improvements through ongoing data scaling. Additionally, we also extend our approach to long chain-of-thought and code reasoning tasks, demonstrating its effectiveness, with details in Appendix C.

2 ScaleQuest: Scaling Question Synthesis from Scratch

2.1 Question Generation from Scratch

The question generation process involves providing only a few prefix tokens from an instruction template (e.g., "`<|begin_of_sentence|>User:`") to guide the model in question generation. A fine-tuned causal language model, which has learned to generate responses based on question-answer pairs (e.g., "`<|begin_of_sentence|>User: {Question}. Assistant: {Response}`"), could potentially be leveraged to generate questions directly (Xu et al., 2024). This is because, during instruction tuning, the model is trained using a causal mask, where each token only attends to preceding tokens. This ensures that the hidden states evolve based on past context without future token influence. However, during instruction tuning, the actual loss is calculated based on the response, i.e.,

$$\mathcal{L} = -\log P(y_i|X, y_{<i}), \tag{1}$$

where $X = \{x_1, x_2, \ldots, x_m\}$ denotes question and $Y = \{y_1, y_2, \ldots, y_n\}$ denotes response. Since $P(x_i|x_{<i})$ is inherently modeled, we need to activate the model's capability for question generation.

2.2 Question Fine-Tuning (QFT)

To activate the model's question generation capability, we first perform Question Fine-Tuning (QFT), where we train the problem-solving model using a small set of problems. To ensure that the generator stops after producing the questions and does not continue generating a response, we add an end-of-sentence token at the end of each question. We use approximately 15K problems (without solutions) by mixing the training set of GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021) datasets as training samples.

The purpose of utilizing these problems is to activate the model's question-generation capability rather than to make the model memorize them. To validate this hypothesis, we train the model separately using the GSM8K and MATH datasets and compare whether the distribution of the generated questions matched that of the training data. To

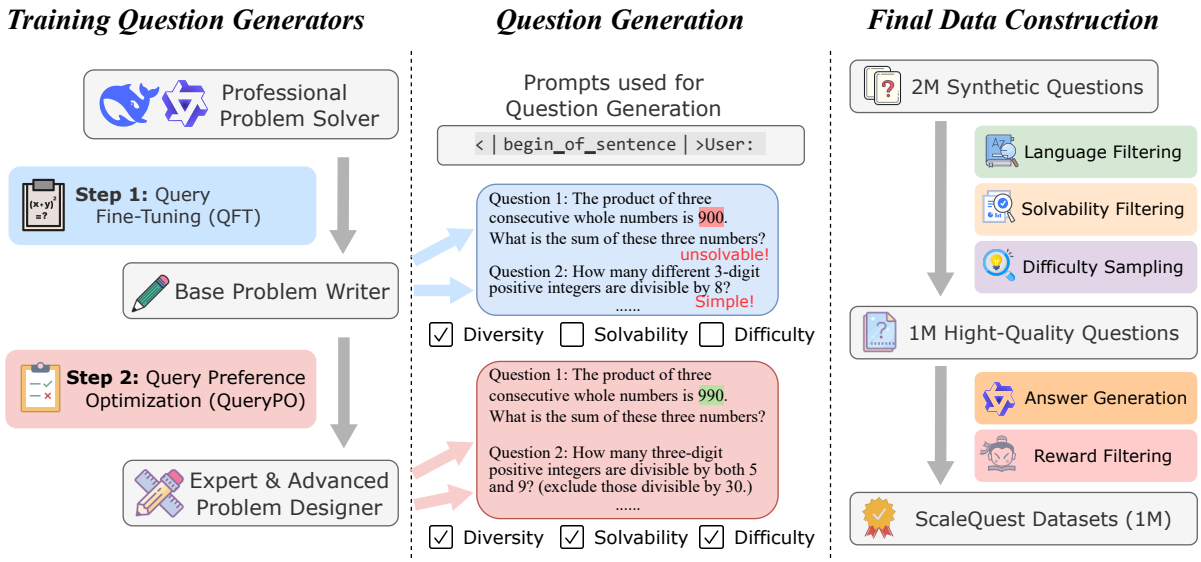

Figure 2: Overview of our ScaleQuest method.

evaluate the question distribution, we use a difficulty classifier, which maps a question into a difficulty score (details in Section 2.4). We perform QFT based on Qwen2-Math-7B-Instruct (Yang et al., 2024a), then use the two QFT models, `Qwen2-QFT-GSM8K` and `Qwen2-QFT-MATH`, to synthesize 10K questions. The difficulty distribution of these four datasets is shown in Figure 3. We find that the generated questions separately differed from both GSM8K and MATH, yet they both converged toward the same distribution. This suggests that the QFT process enhances the model's question-generation capabilities without leading to overfitting the training data.

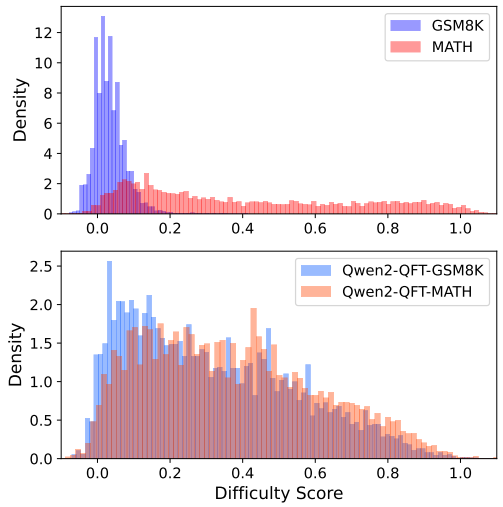

Figure 3: The difficulty distribution of two real-world datasets and two synthetic datasets. The difficulty score is calculated based solely on the problem part.

2.3 Question Preference Optimization (QPO)

The model is able to generate meaningful and diverse questions after QFT, but the quality is still not high enough, as shown in Figure 2. This is reflected in two aspects: (1) solvability: the math problem should have appropriate constraints and correct answers, and (2) difficulty: the model needs to learn from more challenging problems, yet some of the generated questions are still too simple. To address these two aspects, we apply Question Preference Optimization (QPO).

We first use the model after QFT to generate 10K questions. Subsequently, we aim to refine these samples with a primary focus on solvability and difficulty. We leverage an external LLM for optimization as an alternative to manual annotation. However, we find that simultaneously optimizing both poses a challenge for the LLMs. Therefore, for each sample, we randomly select one of the two optimization directions, prioritizing either solvability or difficulty (with optimization prompts in Figure 10 and 11). The optimized questions, denoted as y_w, are treated as preferred data, while the original questions, denoted as y_l, are considered dispreferred data. Inspired by Direct Preference Optimization (DPO) (Rafailov et al., 2024), we propose QPO for question optimization:

$$\mathcal{L}_{\text{QPO}}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(y_w, y_l)\sim\mathcal{D}}\Bigg[\log \sigma\left(\beta \log \frac{\pi_\theta(y_w)}{\pi_{\text{ref}}(y_w)} - \beta \log \frac{\pi_\theta(y_l)}{\pi_{\text{ref}}(y_l)}\right)\Bigg]. \quad (2)$$

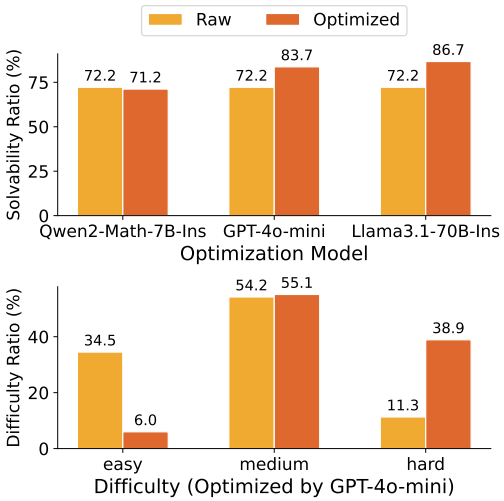

Figure 4: The solvability and difficulty of the raw questions generated by the QFT model and the optimized ones. We use GPT-4o as a manual substitute to evaluate the effectiveness of solvability and difficulty optimization, with evaluation prompt in Figure 12 and 13.

In terms of model selection, we experiment with three models: Qwen2-Math-7B-Instruct, GPT-4o-mini, and Llama3.1-70B-Instruct. The results are shown in Figure 4. In terms of solvability, Qwen2-Math-7B-Instruct proves inadequate for this task, as the optimized questions result in decreased solvability. A possible reason for this is the model's insufficient ability to follow instructions accurately, resulting in many answers that fail to meet the specified optimization constraints.

2.4 Question Filtering

After the QFT and QPO phases, we obtain the question generators Qwen2-Math-QGen. There are still some minor issues in the generated questions, primarily related to language, solvability, and difficulty. To address these challenges, we apply the filtering steps:

Language Filtering The question generator models still produce a substantial number of math questions in other languages, accounting for approximately 20%. Since our focus is on English math questions, we remove non-English questions by identifying questions containing non-English characters and filtering out those samples.

Solvability Filtering Although QPO effectively enhances the solvability of generated questions, some questions remain nonsensical. To filter out such samples, we use Qwen2-Math-7B-Instruct to evaluate whether the question is meaningful and whether the conditions are sufficient, with filtering prompts provided in Figure 12.

Difficulty Sampling We measure the difficulty of a question using the fail rate (Tong et al., 2024) — the proportion of incorrect responses when sampling n responses for a given question. This metric aligns with the intuition that harder questions tend to result in fewer correct responses. Following Tong et al. (2024), we use DeepSeekMath-7B-RL as the sampling model to evaluate the difficulty of each question in the training sets of GSM8K and MATH, obtaining the fail rate for each question as its difficulty score. We then use this data to train a difficulty scorer based on DeepSeekMath-7B-Base with mean squared error (MSE) loss. We then use the scorer to predict the difficulty of each synthetic question. A portion of overly simple questions is then filtered out, which is empirically determined based on the difficulty distribution of the questions.

2.5 Response Generation

We use the reward model score as a metric for evaluating the quality of responses. For each question, we generate 5 solutions and select the solution with the highest reward model scores as the preferred solution. In our experiments, we use InternLM2-7B-Reward (Cai et al., 2024) as our reward model. This choice is primarily guided by the model's performance on the reasoning subset of the Reward Bench (Lambert et al., 2024).

3 Experiment

3.1 Experimental Setup

Training Problem Designers In addition to Qwen2-Math-QGen, we also train another question generator based on DeepSeekMath-7B-RL. We find that combining questions synthesized by multiple generators enhances data diversity, leading to improved final performance, as further discussed in section 3.3. In this process of the overall data synthesis, many models are mentioned in the context of QFT and QPO. We explain the inherent insights behind the model selection in Appendix D.

Question & Response Generation The two question generation models are then utilized to generate a total of 2 million questions, with 1 million from each model. For difficulty filtering, we filter out a portion of overly simple questions generated by `DeepSeekMath-QGen` and keep the questions generated by `Qwen2-Math-QGen` as the difficulty

Model	Synthesis Model	GSM8K	MATH	College Math	Olympiad Bench	Average
Teacher Models in Data Synthesis						
GPT-4-0314	-	94.7	52.6	24.4	-	-
GPT-4-Turbo-24-04-09	-	94.5	73.4	-	-	-
GPT-4o-2024-08-06	-	92.9	81.1	50.2	43.3	66.9
DeepSeekMath-7B-RL	-	88.2	52.4	41.4	19.0	49.3
Qwen2-Math-7B-Instruct	-	89.5	73.1	50.5	37.8	62.7
General Base Model						
Mistral-7B-WizardMath	GPT-4	81.9	33.3	21.5	8.6	36.3
Mistral-7B-MetaMath	GPT-3.5	77.7	28.2	19.1	5.8	32.7
Mistral-7B-MMIQC	GPT-4	75.7	36.3	24.8	10.8	36.9
Mistral-7B-MathScale	GPT-3.5	74.8	35.2	21.8	-	-
Mistral-7B-KPMath	GPT-4	82.1	46.8	-	-	-
Mistral-7B-DART-Math	DSMath-7B-RL	81.1	45.5	29.4	14.7	42.7
Mistral-7B-NuminaMath	GPT-4o	82.1	49.4	33.8	19.4	46.2
Mistral-7B-ScaleQuest	Qwen2-Math-7B-Ins	**88.5**	**62.9**	**43.5**	**26.8**	**55.4**
Llama3-8B-MetaMath	GPT-3.5	77.3	32.5	20.6	5.5	34.0
Llama3-8B-MMIQC	GPT-4	77.6	39.5	29.5	9.6	39.1
Llama3-8B-DART-Math	DSMath-7B-RL	81.1	46.6	28.8	14.5	42.8
Llama3-8B-NuminaMath	GPT-4o	77.2	50.7	33.2	17.8	44.7
Llama3-8B-ScaleQuest	Qwen2-Math-7B-Ins	**87.9**	**64.4**	**42.8**	**25.3**	**55.1**
Math-Specialized Base Model						
DeepSeekMath-7B-Instruct	-	82.7	46.9	37.1	14.2	45.2
DeepSeekMath-7B-MMIQC	GPT-4	79.0	45.3	35.3	13.0	43.2
DeepSeekMath-7B-KPMath-Plus	GPT-4	83.9	48.8	-	-	-
DeepSeekMath-7B-DART-Math	DSMath-7B-RL	86.8	53.6	40.7	21.7	50.7
DeepSeekMath-7B-Numina-Math	GPT-4o	75.4	55.2	36.9	19.9	46.9
DeepSeekMath-7B-ScaleQuest	Qwen2-Math-7B-Ins	**89.5**	**66.6**	**47.7**	**29.9**	**58.4**
Qwen2-Math-7B-MetaMath	GPT-3.5	83.9	49.5	39.9	17.9	47.8
Qwen2-Math-7B-DART-Math	DSMath-7B-RL	88.6	58.8	45.4	23.1	54.0
Qwen2-Math-7B-Numina-Math	GPT-4o	84.6	65.6	45.5	33.6	57.3
Qwen2-Math-7B-ScaleQuest	Qwen2-Math-7B-Ins	**89.7**	**73.4**	**50.0**	**38.5**	**62.9**

Table 1: Main results on four mathematical reasoning benchmarks. **Bold** means the best score with the same base model. The baselines use different synthesis models for both question synthesis and response generation, such as GPT-3.5, GPT-4, and GPT-4o. For our approach, DSMath-7B-QGen and Qwen2-Math-7B-QGen are utilized for question synthesis, with Qwen2-Math-7B-Instruct used for response generation. If multiple models are used, only the latest released one is marked. More details about these datasets are shown in Table 7.

distribution is more balanced. Based on the problems, we synthesize responses (section 2.5) using Qwen2-Math-7B-Instruct (Yang et al., 2024a). For each problem, we sample 5 solutions and select the one with the highest reward score as the final response. The final dataset consists of 1 million problem-solution pairs. A detailed analysis of our constructed dataset is provided in Appendix B.

Instruction Tuning We conduct instruction tuning on the synthetic problems and solutions using two general base models, Mistral-7B (Jiang et al., 2023) and Llama3-8B (Dubey et al., 2024), as well as two math-specialized base models, DeepSeekMath-7B (Shao et al., 2024) and Qwen2-Math-7B (Yang et al., 2024a). More hyperparameters in the process are provided in Appendix A.

Evaluation and Metrics We assess the fine-tuned models' performance across four datasets of increasing difficulty. Along with the widely used GSM8K (elementary level) and MATH (competition level), we include two more challenging benchmarks: College Math (Tang et al., 2024) (college level) and Olympiad Bench (He et al., 2024) (Olympiad level). For evaluation, we employ the script from Tong et al. (2024) to extract final answers and determine correctness by comparing answer equivalency. The generated outputs are all in the form of natural language Chain-of-Thought (CoT) reasoning (Wei et al., 2022) through greedy decoding, with no tool integration, and we report

zero-shot pass@1 accuracy. We also conduct more comprehensive evaluation, with details shown in Appendix E. We also extend our approach to the long cot and code reasoning task, with details and experiments provided in Appendix C.

Compared Baselines We mainly compare with data synthesis methods, including question-driven approach such as WizardMath (Luo et al., 2023), MetaMath (Yu et al., 2023a), MMIQC (Liu and Yao, 2024), Orca-Math (Mitra et al., 2024) and knowledge-driven approach such as KP-Math (Huang et al., 2024a), MathScale (Tang et al., 2024). In addition to this, we also involve other large math corpus like DART-Math (Tong et al., 2024) and Numina-Math (Li et al., 2024c). More details of these datasets are shown in Table 7.

3.2 Main Results

ScaleQuest significantly outperforms other baselines Table 1 presents the results. ScaleQuest significantly outperforms previous synthetic methods, with average performance improvements ranging from 5.6% to 11.5% over the prior state-of-the-art (SoTA) on both general base models and math-specialized foundation models. Qwen2-Math-7B-ScaleQuest achieve a zero-shot pass@1 accuracy of 73.4 on the MATH benchmark, matching the performance of GPT-4-Turbo. For out-of-domain tasks, Qwen2-Math-7B-ScaleQuest outperform its teacher model, Qwen2-Math-7B-Instruct, with scores of 89.7 on the GSM8K benchmark, 73.4 on the MATH benchmark, and 38.5 on the Olympiad benchmark. In this experiment, we do not strictly control for identical training data volumes due to practical constraints (i.e., some datasets are not publicly available). To address this, we compare the results with publicly available datasets under equal data volumes in Appendix E.2. It's important to highlight that Qwen2-Math-7B-Instruct has undergone preference alignment, utilizing the powerful reward model Qwen2-Math-RM-72B (Yang et al., 2024a), while our model is only an instruction tuning version.

ScaleQuest scales well with increasing data We also compare the scalability of our dataset with other publicly available datasets, including MetaMath (Yu et al., 2023a), DART-Math (Tong et al., 2024), and Numina-Math (Li et al., 2024c). We train Llama3-8B on these datasets and compare their performance changes with increased data size. The results are presented in Figure 1. For the in-domain evaluation (MATH), our method demonstrates high data efficiency, achieving superior results with the same amount of data. In out-of-domain evaluations (Olympiad Bench), it also shows strong scalability, continuing to improve even as other datasets reach their limits. A limited question set leads to constrained improvements in model performance, as demonstrated by the results of DART-Math, which relies on a small number of questions and generates numerous correct answers through rejection sampling. Our results further demonstrate that diverse questions support sustained performance growth, emphasizing the need for broader and more varied question generation.

Question Source	GSM8K	MATH	CM	OB	**Avg**
MetaMath	84.5	53.8	40.1	22.1	50.1
OrcaMath	84.2	53.7	40.5	23.7	50.5
NuminaMath	86.0	65.9	46.1	**30.2**	57.1
ScaleQuest	**89.5**	**66.6**	**47.7**	29.9	**58.4**

Table 2: Comparison of question quality across different datasets. To ensure consistency, all responses are generated using Qwen2-Math-7B-Instruct with the same reward filtering process. CM and OB refer to College Math and Olympiad Bench, respectively. Additional results based on the other three backbone models are shown in Appendix E.4.

3.3 Ablation study

Each sub-method contributes to the final performance To validate the effectiveness of each of our sub-methods, including QFT, QPO, and reward filtering, we conduct an ablation study. We evaluate the quality of the questions generated by the models across three dimensions: solvability, difficulty, and performance in instruction tuning. For solvability and difficulty, we use the same evaluation process as stated in section 2.3.

The results are shown in Figure 5. The "raw model" refers to using the instruct model to directly generate instructions and responses, as done in Xu et al. (2024). To ensure fairness, we also generate 1M question-response pairs using their method based on Qwen2-Math-7B-Instruct, which are used to train Llama3-8B-Base. After applying QFT and QPO, the model's performance improves across all three evaluation dimensions, demonstrating the effectiveness of each sub-method.

Comparison of question quality To directly compare the question quality of our constructed data with other open-source datasets, we use the

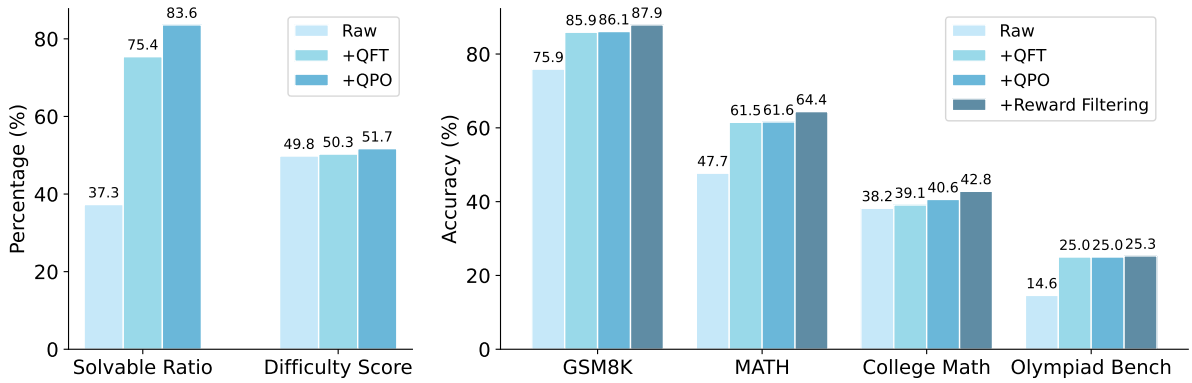

Figure 5: A comparison of the synthetic dataset generated by the raw instruct model, the model after QFT, the model after QPO, and the final dataset after applying reward filtering. **Left:** The solvable ratio and difficulty score of the generated questions. The solvable ratio refers to the proportion of generated questions that are judged as "solvable", while the difficulty score represents the average difficulty rating assigned to each generated question. For difficulty evaluation, we calculate the dataset's average difficulty score based on ratings for each question: "very easy" is rated as 20 points, "easy" as 40 points, "medium" as 60 points, "hard" as 80 points, and "very hard" as 100 points. **Right**: The instruction tuning effectiveness on Llama3-8B-Base.

Dataset	GSM8K	MATH	CM	OB	**Avg**
SQ-DSMath	87.6	52.2	39.8	19.4	49.8
SQ-Qwen2	86.8	56.1	39.6	18.7	50.3
Mixed	**87.8**	**58.0**	**40.1**	**22.2**	**52.0**

Table 3: The performance of Mistral-7B fine-tuned on ScaleQuest-DSMath, ScaleQuest-Qwen2, and a mix of both. We fix the training data size at 400K and find that the mixed data results in the largest improvement.

same model, Qwen2-Math-7B-Instruct, to generate responses. We then fine-tune DeepSeekMath-7B-Base using these synthetic datasets. As shown in Table 2, our model outperforms other synthetic datasets like MetaMath and OrcaMath, highlighting the high quality of our questions. NuminaMath also demonstrates competitive performance, largely due to the fact that many of its questions are drawn from real-world scenarios. This highlights the importance of question quality for synthetic data.

Multiple question generators enhance data diversity We use two models as question generators: DSMath-QGen and Qwen2-Math-QGen, which are based on DeepSeekMath (Shao et al., 2024) and Qwen2-Math (Yang et al., 2024a), respectively. To explore the impact of using multiple question generators, we compare the effects of using data synthesized by a single generator versus a mix of data from both. As shown in Table 3, we find that the mixed data outperforms the data generated by either single generator. A possible explanation for this improvement is the increased data diversity. In fact, we observe that DSMath-QGen tends to generate simpler, more real-world-oriented questions, while Qwen2-Math-QGen produces more challenging, theory-driven ones.

3.4 Human Evaluation Results

We conduct a human evaluation of the generated data, focusing on three aspects: clarity, reasonableness, and real-world relevance. For reference, we also include two high-quality, human-curated datasets, GSM8K and MATH. A total of 40 examples are sampled from each dataset and evaluated based on clarity, coherence, and real-world relevance, with scores ranging from 1 to 5. The results are presented in Table 5. In terms of clarity and reasonableness, our synthetic data surpasses NuminaMath but still falls short of the high-quality, real-world datasets like the training sets of GSM8K and MATH. Regarding real-world relevance, GSM8K leans toward practical, real-life scenarios, while MATH focuses more on theoretical mathematical derivations. Our generated data can be seen as a balance between the two.

3.5 Cost Analysis

The data synthesis process is conducted on a server with 8 A100-40G-PCIe GPUs. We summarize our overall costs in Table 4. Generating 1 million data samples requires only 522.9 GPU hours (approximately 2.7 days on an 8-GPU server), with an estimated cost of $680.8 for cloud server rental.[2]

[2] https://lambdalabs.com/service/gpu-cloud

Phase		Type	# Samples	GPU hours	Cost ($)
QFT	Training DSMath-QFT	Train	15K	2.0	2.6
	Training Qwen2-Math-QFT	Train	15K	1.9	2.5
QPO	Generate Questions	Infer	10K×2	0.4	0.5
	Construct Preference Data	API	10K×2	-	6.2
	QPO Training	Train	10K×2	6.6	8.5
Data Synthesis	Question Generation	Infer	2M	38.4	49.5
	solvability & difficulty check	Infer	2M	110.6	142.7
	Response Generation	Infer	1M×5	251.0	323.8
	Reward Scoring	Infer	1M×5	112.0	144.5
Total			1M	522.9	680.8
GPT-4 cost (generating the same number of tokens)			-	-	24,939.5
GPT-4o cost (generating the same number of tokens)			-	-	6,115.9

Table 4: Cost analysis of the entire data synthesis process. We also estimate the cost of generating the same number of tokens using proprietary models GPT-4 and GPT-4o for comparison.

Dataset	Clarity	Reasonableness	Real-world relevance
GSM8K	4.4	4.5	3.9
MATH	4.1	4.3	2.4
NuminaMath	3.8	4.0	2.4
ScaleQuest	3.9	4.0	2.8

Table 5: Human Evaluation Results.

This is only about 10% of the cost of generating the same data using GPT-4o, demonstrating the cost-effectiveness of our method.

4 Related Work

4.1 Mathematical Reasoning

Solving math problems is regarded as a key measure of evaluating the reasoning ability of LLMs. Recent advancements in mathematical reasoning for LLMs, including models like OpenAI o1, Claude-3.5, Gemini (Reid et al., 2024), DeepSeek-Math (Shao et al., 2024), InternLM2-Math (Cai et al., 2024), and Qwen2.5-Math (Yang et al., 2024b), have spurred the development of various approaches to improve reasoning capabilities of LLMs on math-related tasks. To strengthen the math reasoning capabilities of LLMs, researchers have focused on areas such as prompting techniques (Chia et al., 2023; Chen et al., 2023; Zhang et al., 2023), data construction for pretraining (Lewkowycz et al., 2022; Azerbayev et al., 2023; Zhou et al., 2024; Shao et al., 2024) and instruction tuning (Luo et al., 2023; Yue et al., 2023), tool-integrated reasoning(Chen et al., 2022; Gao et al., 2023; Gou et al., 2023; Wang et al., 2023; Yue et al., 2024; Yin et al., 2024; Zhang et al., 2024), and preference tuning (Ma et al., 2023; Luong et al., 2024; Shao et al., 2024; Lai et al., 2024). Our work primarily focuses on math data synthesis for instruction tuning.

4.2 Data Synthesis for Instruction Tuning

High-quality reasoning data, particularly well-crafted questions, is in short supply. Prior efforts have mostly started with a small set of human-annotated seed instructions and expanded them through few-shot prompting. We categorize them into two types: question-driven augmentation (Luo et al., 2023; Yu et al., 2023a; Li et al., 2024a; Liu and Yao, 2024; Li et al., 2024b) and knowledge-driven augmentation (Didolkar et al., 2024; Shah et al., 2024; Tang et al., 2024; Huang et al., 2024a). There are other methods for enhancing dataset quality as well. DART-Math (Tong et al., 2024) focuses on enhancing the quality of responses by using difficulty-aware rejection sampling. In contrast, Numina-Math (Li et al., 2024c) improves its dataset by collecting more real-world and synthetic data. These high-quality datasets can be integrated with our constructed dataset, resulting in an improved data mix for more effective instruction tuning.

5 Conclusion

In this work, we propose ScaleQuest, a novel data synthesis framework that unlocks the ability of lightweight models to independently generate large-scale, high-quality reasoning data from scratch, at a low cost. Using this dataset, we fine-tune the model and achieve remarkable improvements, with gains ranging from 29.2% to 46.4% compared to the base model, outperforming the strong baselines.

Limitation

We explore the potential of lightweight open-sourced models to generate high-quality instruction tuning data. However, the effectiveness on larger and more powerful models, such as Qwen2.5-Math-72B-Instruct and Llama3.3-70B-Instruct, remains uncertain. Our future research will focus on experimenting with these larger models. Additionally, although the optimizations for solvability and difficulty have shown positive effects, the model-generated responses still fall short of being fully satisfactory. Therefore, further improvements in question preference alignment are crucial and will be an important direction for our future work.

Acknowledgements

We want to thank all the anonymous reviewers for their valuable comments. This work was supported by the National Science Foundation of China (NSFC No. 62206194), the Natural Science Foundation of Jiangsu Province, China (Grant No. BK20220488), the Young Elite Scientists Sponsorship Program by CAST (2023QNRC001), the Priority Academic Program Development of Jiangsu Higher Education Institutions, and Sponsored by CCF-Tencent Rhino-Bird Open Research Fund. We also acknowledge MetaStone Tech. Co. for providing us with the software, optimisation on high performance computing and computational resources required by this work.

References

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*.

Tom B Brown. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Zheng Cai, Maosong Cao, Haojiong 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*.

Jiaao Chen, Xiaoman Pan, Dian Yu, Kaiqiang Song, Xiaoyang Wang, Dong Yu, and Jianshu Chen. 2023. Skills-in-context prompting: Unlocking compositionality in large language models. *arXiv preprint arXiv:2308.00304*.

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*.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W Cohen. 2022. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *arXiv preprint arXiv:2211.12588*.

Yew Ken Chia, Guizhen Chen, Luu Anh Tuan, Soujanya Poria, and Lidong Bing. 2023. Contrastive chain-of-thought prompting. *arXiv preprint arXiv:2311.09277*.

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*.

Aniket Didolkar, Anirudh Goyal, Nan Rosemary Ke, Siyuan Guo, Michal Valko, Timothy Lillicrap, Danilo Rezende, Yoshua Bengio, Michael Mozer, and Sanjeev Arora. 2024. Metacognitive capabilities of llms: An exploration in mathematical problem solving. *arXiv preprint arXiv:2405.12205*.

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*.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Pal: Program-aided language models. In *International Conference on Machine Learning*, pages 10764–10799. PMLR.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yujiu Yang, Minlie Huang, Nan Duan, Weizhu Chen, et al. 2023. Tora: A tool-integrated reasoning agent for mathematical problem solving. *arXiv preprint arXiv:2309.17452*.

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, Yu Wu, 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. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. *arXiv preprint arXiv:2402.14008*.

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. *arXiv preprint arXiv:2103.03874*.

Yiming Huang, Xiao Liu, Yeyun Gong, Zhibin Gou, Yelong Shen, Nan Duan, and Weizhu Chen. 2024a. Key-point-driven data synthesis with its enhancement on mathematical reasoning. *arXiv preprint arXiv:2403.02333*.

Yinya Huang, Xiaohan Lin, Zhengying Liu, Qingxing Cao, Huajian Xin, Haiming Wang, Zhenguo Li, Linqi Song, and Xiaodan Liang. 2024b. Mustard: Mastering uniform synthesis of theorem and proof data. *arXiv preprint arXiv:2402.08957*.

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*.

Hakan Inan, Kartikeya Upasani, Jianfeng Chi, Rashi Rungta, Krithika Iyer, Yuning Mao, Michael Tontchev, Qing Hu, Brian Fuller, Davide Testuggine, et al. 2023. Llama guard: Llm-based input-output safeguard for human-ai conversations. *arXiv preprint arXiv:2312.06674*.

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*.

Mario Michael Krell, Matej Kosec, Sergio P Perez, and Andrew Fitzgibbon. 2021. Efficient sequence packing without cross-contamination: Accelerating large language models without impacting performance. *arXiv preprint arXiv:2107.02027*.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

Xin Lai, Zhuotao Tian, Yukang Chen, Senqiao Yang, Xiangru Peng, and Jiaya Jia. 2024. Step-dpo: Step-wise preference optimization for long-chain reasoning of llms. *arXiv preprint arXiv:2406.18629*.

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.

Chen Li, Weiqi Wang, Jingcheng Hu, Yixuan Wei, Nanning Zheng, Han Hu, Zheng Zhang, and Houwen Peng. 2024a. Common 7b language models already possess strong math capabilities. *arXiv preprint arXiv:2403.04706*.

Haoran Li, Qingxiu Dong, Zhengyang Tang, Chaojun Wang, Xingxing Zhang, Haoyang Huang, Shaohan Huang, Xiaolong Huang, Zeqiang Huang, Dongdong Zhang, et al. 2024b. Synthetic data (almost) from scratch: Generalized instruction tuning for language models. *arXiv preprint arXiv:2402.13064*.

Jia Li, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Huang, Kashif Rasul, Longhui Yu, Albert Q Jiang, Ziju Shen, et al. 2024c. Numinamath: The largest public dataset in ai4maths with 860k pairs of competition math problems and solutions. *Hugging Face repository*.

Zhenwen Liang, Dian Yu, Wenhao Yu, Wenlin Yao, Zhihan Zhang, Xiangliang Zhang, and Dong Yu. 2024. Mathchat: Benchmarking mathematical reasoning and instruction following in multi-turn interactions. *arXiv preprint arXiv:2405.19444*.

Haoxiong Liu and Andrew Chi-Chih Yao. 2024. Augmenting math word problems via iterative question composing. *arXiv preprint arXiv:2401.09003*.

Zimu Lu, Aojun Zhou, Houxing Ren, Ke Wang, Weikang Shi, Junting Pan, Mingjie Zhan, and Hongsheng Li. 2024. Mathgenie: Generating synthetic data with question back-translation for enhancing mathematical reasoning of llms. *arXiv preprint arXiv:2402.16352*.

Haipeng Luo, Qingfeng Sun, Can Xu, Pu Zhao, Jianguang Lou, Chongyang Tao, Xiubo Geng, Qingwei Lin, Shifeng Chen, and Dongmei Zhang. 2023. Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct. *arXiv preprint arXiv:2308.09583*.

Trung Quoc Luong, Xinbo Zhang, Zhanming Jie, Peng Sun, Xiaoran Jin, and Hang Li. 2024. Reft: Reasoning with reinforced fine-tuning. *arXiv preprint arXiv:2401.08967*.

Qianli Ma, Haotian Zhou, Tingkai Liu, Jianbo Yuan, Pengfei Liu, Yang You, and Hongxia Yang. 2023. Let's reward step by step: Step-level reward model as the navigators for reasoning. *arXiv preprint arXiv:2310.10080*.

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*.

Philipp Moritz, Robert Nishihara, Stephanie Wang, Alexey Tumanov, Richard Liaw, Eric Liang, Melih Elibol, Zongheng Yang, William Paul, Michael I Jordan, et al. 2018. Ray: A distributed framework for emerging {AI} applications. In *13th USENIX symposium on operating systems design and implementation (OSDI 18)*, pages 561–577.

Eirini Ntoutsi, Pavlos Fafalios, Ujwal Gadiraju, Vasileios Iosifidis, Wolfgang Nejdl, Maria-Esther Vidal, Salvatore Ruggieri, Franco Turini, Symeon Papadopoulos, Emmanouil Krasanakis, et al. 2020. Bias in data-driven artificial intelligence systems—an introductory survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 10(3):e1356.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36.

Machel Reid, Nikolay Savinov, Denis Teplyashin, Dmitry Lepikhin, Timothy Lillicrap, Jean-baptiste Alayrac, Radu Soricut, Angeliki Lazaridou, Orhan Firat, Julian Schrittwieser, et al. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*.

Vedant Shah, Dingli Yu, Kaifeng Lyu, Simon Park, Nan Rosemary Ke, Michael Mozer, Yoshua Bengio, Sanjeev Arora, and Anirudh Goyal. 2024. Ai-assisted generation of difficult math questions. *arXiv preprint arXiv:2407.21009*.

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*.

Zhengyang Tang, Xingxing Zhang, Benyou Wan, and Furu Wei. 2024. Mathscale: Scaling instruction tuning for mathematical reasoning. *arXiv preprint arXiv:2403.02884*.

Yuxuan Tong, Xiwen Zhang, Rui Wang, Ruidong Wu, and Junxian He. 2024. Dart-math: Difficulty-aware rejection tuning for mathematical problem-solving. *arXiv preprint arXiv:2407.13690*.

Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(11).

Ke Wang, Houxing Ren, Aojun Zhou, Zimu Lu, Sichun Luo, Weikang Shi, Renrui Zhang, Linqi Song, Mingjie Zhan, and Hongsheng Li. 2023. Mathcoder: Seamless code integration in llms for enhanced mathematical reasoning. *arXiv preprint arXiv:2310.03731*.

Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2021. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*.

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.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. 2023. Wizardlm: Empowering large language models to follow complex instructions. *arXiv preprint arXiv:2304.12244*.

Zhangchen Xu, Fengqing Jiang, Luyao Niu, Yuntian Deng, Radha Poovendran, Yejin Choi, and Bill Yuchen Lin. 2024. Magpie: Alignment data synthesis from scratch by prompting aligned llms with nothing. *arXiv preprint arXiv:2406.08464*.

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*.

Shuo Yin, Weihao You, Zhilong Ji, Guoqiang Zhong, and Jinfeng Bai. 2024. Mumath-code: Combining tool-use large language models with multi-perspective data augmentation for mathematical reasoning. *arXiv preprint arXiv:2405.07551*.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2023a. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*.

Weichen Yu, Tianyu Pang, Qian Liu, Chao Du, Bingyi Kang, Yan Huang, Min Lin, and Shuicheng Yan. 2023b. Bag of tricks for training data extraction from language models. In *International Conference on Machine Learning*, pages 40306–40320. PMLR.

Xiang Yue, Xingwei Qu, Ge Zhang, Yao Fu, Wenhao Huang, Huan Sun, Yu Su, and Wenhu Chen. 2023. Mammoth: Building math generalist models through hybrid instruction tuning. *arXiv preprint arXiv:2309.05653*.

Xiang Yue, Tuney Zheng, Ge Zhang, and Wenhu Chen. 2024. Mammoth2: Scaling instructions from the web. *arXiv preprint arXiv:2405.03548*.

Weihao Zeng, Can Xu, Yingxiu Zhao, Jian-Guang Lou, and Weizhu Chen. 2024. Automatic instruction evolving for large language models. *arXiv preprint arXiv:2406.00770*.

Beichen Zhang, Kun Zhou, Xilin Wei, Xin Zhao, Jing Sha, Shijin Wang, and Ji-Rong Wen. 2024. Evaluating and improving tool-augmented computation-intensive math reasoning. *Advances in Neural Information Processing Systems*, 36.

Yifan Zhang, Jingqin Yang, Yang Yuan, and Andrew Chi-Chih Yao. 2023. Cumulative reasoning with large language models. *arXiv preprint arXiv:2308.04371*.

Wenting Zhao, Xiang Ren, Jack Hessel, Claire Cardie, Yejin Choi, and Yuntian Deng. 2024. Wildchat: 1m chatgpt interaction logs in the wild. *arXiv preprint arXiv:2405.01470*.

Tianyu Zheng, Ge Zhang, Tianhao Shen, Xueling Liu, Bill Yuchen Lin, Jie Fu, Wenhu Chen, and Xiang Yue. 2024. Opencodeinterpreter: Integrating code generation with execution and refinement. *arXiv preprint arXiv:2402.14658*.

Kun Zhou, Beichen Zhang, Jiapeng Wang, Zhipeng Chen, Wayne Xin Zhao, Jing Sha, Zhichao Sheng, Shijin Wang, and Ji-Rong Wen. 2024. Jiuzhang3. 0: Efficiently improving mathematical reasoning by training small data synthesis models. *arXiv preprint arXiv:2405.14365*.

Terry Yue Zhuo, Minh Chien Vu, Jenny Chim, Han Hu, Wenhao Yu, Ratnadira Widyasari, Imam Nur Bani Yusuf, Haolan Zhan, Junda He, Indraneil Paul, et al. 2024. Bigcodebench: Benchmarking code generation with diverse function calls and complex instructions. *arXiv preprint arXiv:2406.15877*.

A Hyper-parameters settings

Training Problem Designers During the QFT stage, both models are trained on a mixed training subset of GSM8K and MATH problems, containing a total of 15K problems. We train for only 1 epoch, considering that training for more epochs might cause the models to overfit the training problems and negatively impact the diversity of generated questions. We also use sequence packing (Krell et al., 2021) to accelerate training. In the QPO stage, we use 10K preference data for training, with a learning rate of 5e-7 and a batch size of 128.

Question Generation During this process, we set the maximum generation length to 512, a temperature of 1.0, and a top-p value of 0.99. To ensure quality, we apply a question filtering pipeline (section 2.4) that involves language filtering, solvability filtering, and difficulty sampling. This process refines the dataset, leaving approximately 1M questions, 400K from `DeepSeek-QGen` and 600K from `Qwen2-Math-QGen`.

Response Generation In the process, we set the maximum generation length to 2048, with a temperature of 0.7 and top-p of 0.95. We use chain-of-thought prompt (Wei et al., 2022) to synthesize solutions. We use vLLM (Kwon et al., 2023) to accelerate the generation and Ray (Moritz et al., 2018) to deploy distributed inference.

Instruction Tuning All models are fine-tuned for 3 epochs in our experiments unless specified otherwise. We use a linear learning rate schedule with a 3% warm-up ratio, reaching a peak of 5e-5 for Llama3 and DeepSeekMath and 1e-5 for the other models, followed by cosine decay to zero.

B Additional Data Statistics

Filtering process The entire data generation process is illustrated in Figure 6. After using the two question generators to produce 2 million questions from scratch, we perform a filtering process, including language filtering, solvability checks, and difficulty sampling. These steps filter out 20.1%, 19.4%, and 9.2% of the samples, respectively, resulting in a final question set of 1 million questions. In the subsequent response generation process, we filter out responses without answers by checking for key phrases such as "The answer is" or "\boxed{}". This step eliminates a negligible portion of the samples, as most of the filter questions are solvable and did not pose any confusion for the response generation model.

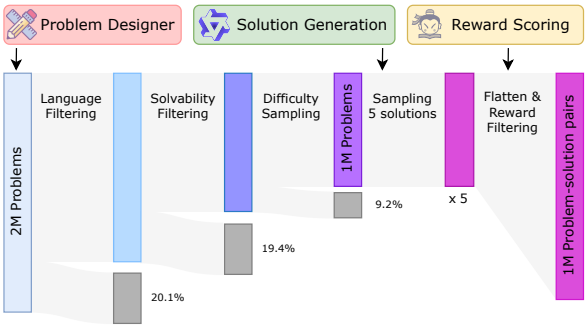

Figure 6: Overview of our filtering process.

Dataset Coverage We analyze the dataset coverage through two aspects: (1) Problem Topic Coverage, such as algebra and geometry. Following Huang et al. (2024a), we use GPT-4o to categorize the topics of the given questions, with prompt illustrated in Figure 14. Figure 7 presents the results. We find that the topics cover the major areas of mathematics, such as arithmetic, algebra, geometry, and others. (2) Embedding space analysis. Following Zhao et al. (2024) and Xu et al. (2024), we first compute the input embeddings of the questions and then project them into a two-dimensional space using t-SNE (Van der Maaten and Hinton, 2008). We include only real-world datasets, such as GSM8K (Cobbe et al., 2021), MATH (Hendrycks et al., 2021), and NuminaMath (Li et al., 2024c) (which contains a small portion of synthetic questions). As shown in Figure 8, our synthetic data closely resembles the real-world questions.

Data Leakage Analysis We conduct an n-gram similarity analysis between the generated questions and all test sets from both our dataset and other baseline datasets. Based on prior empirical analysis (Brown, 2020; Wei et al., 2021), we set n=13 to prevent spurious collisions and calculate how much the test sets overlap with training data to assess data contamination. Table 6 shows the clean ratio of our dataset and other baseline datasets. The results demonstrate that our dataset achieves a relatively high level of data cleanliness compared to other datasets, suggesting that our method generates novel questions instead of memorizing existing ones.

Safety Analysis We use Llama3-8B-Guard (Inan et al., 2023) as a discriminator model to detect any unsafe elements in the data. After sampling 10K

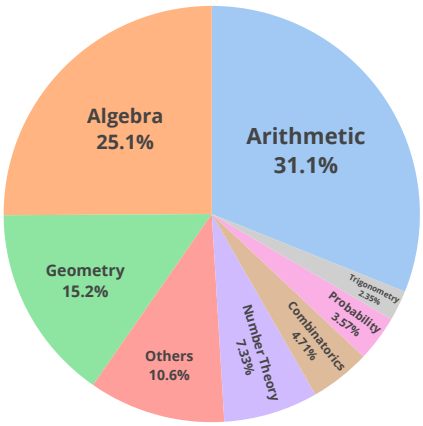

Figure 7: Topic distribution of our generated dataset.

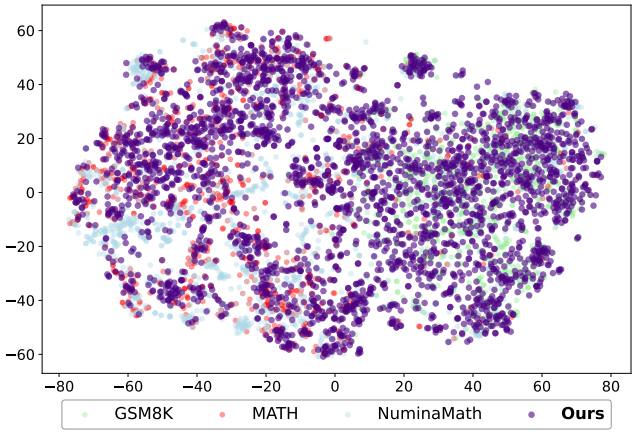

Figure 8: t-SNE plot of our dataset, with GSM8K, MATH, and NuminaMath.

Dataset	GSM8K	MATH	CM	OB	Avg
MetaMath	99.8	92.2	100	99.7	97.9
NuminaMath	99.8	89.8	99.9	86.8	94.1
DART-Math	99.8	91.5	100.0	99.6	97.7
MMIQC	99.8	88.0	98.9	97.9	96.2
SQ (Ours)	99.9	92.8	99.8	97.2	97.4

Table 6: Overlap statistics for the datasets used. We report the clean ratio of the test set, representing the percentage of test samples that have no matching n-grams with samples in the training set.

instances from the 1 million samples, we find that only 0.1% are flagged as unsafe.

Generated Examples We sample several generated examples from our datasets, as shown in Figure 17, 18 and 19. The generated math problems are of high quality, driving effective learning.

C ScaleQuest for Broader tasks

C.1 Long Chain-of-Thought Mathematical Reasoning

DeepSeek-R1 (Guo et al., 2025) demonstrated the effectiveness of Long Chain-of-Thought (CoT) reasoning in tackling challenging mathematical reasoning tasks. In this part, we further investigate the capability of ScaleQuest to generate high-quality long CoT data.

Settings We utilize the latest Qwen2.5-Math-7B model (Yang et al., 2024b) for question fine-tuning and preference optimization. To construct long CoT data, we select DeepSeek-R1-Distill-Qwen-7B as the response generator. The rest of the pipeline remains unchanged from prior work. In total, we synthesized 150K Long Chain-of-Thought training samples.

Results We then fine-tuned DeepSeek-R1-Distill-Qwen-7B on this dataset, which also serves as the teacher model during data synthesis. Table 8 presents the results, showing that the fine-tuned model outperforms its teacher, demonstrating a strong potential for self-improvement.

C.2 Code Reasoning

We also extend our ScaleQuest method to the Code Reasoning Task as a simple validation. We made the following modifications to adapt to the code reasoning task:

Settings We choose DeepSeek-Coder-7B-Instruct (Guo et al., 2024) and Qwen2.5-Coder-7B-Instruct (Hui et al., 2024) as two problem-solving models to perform question fine-tuning on 20K questions randomly sampled from CodeFeedBack (Zheng et al., 2024). For Question Preference Optimization, we also focus on solvability and difficulty, making slight modifications to the prompts based on the code reasoning task. Our evaluation cover HumanEval (Chen et al., 2021), MBPP (Austin et al., 2021), and BigCodeBench (Zhuo et al., 2024), using the same evaluation script as Qwen2.5-Coder. We report pass@1 results using greedy search.

Results The results are presented in Table 9. Compared to the widely used refined version of CodeFeedback, namely CodeFeedback-Filtered, our generated data outperforms it, with an average improvement of 5.9 across the three baselines. Additionally, we enhance the Response portion of

Dataset	Size	Synthesis Model	Public
WizardMath (Luo et al., 2023)	96K	GPT-4	✗
MetaMath (Yu et al., 2023a)	395K	GPT-3.5-Turbo	✓
MMIQC (Liu and Yao, 2024)	2294K	GPT-4 & GPT-3.5-Turbo & Human	✓
Orca-Math (Mitra et al., 2024)	200K	GPT-4-Turbo	✓
Xwin-Math (Li et al., 2024a)	1440K	GPT-4-Turbo	✗
KPMath-Plus (Huang et al., 2024a)	1576K	GPT-4	✗
MathsScale (Tang et al., 2024)	2021K	GPT-3.5 & Human	✗
DART-Math (Tong et al., 2024)	585K	DeepSeekMath-7B-RL	✓
Numina-Math (Li et al., 2024c)	860K	GPT-4 & GPT-4o	✓
ScaleQuest	1000K	DeepSeekMath-7B-RL Qwen2-Math-7B-Instruct	✓

Table 7: Comparison between our constructed dataset and previous datasets.

Model	GSM8k	MATH500	AIME24	AMC25	Average
DS-R1-Distill-Qwen-7B	91.5	90.2	43.3	75.0	75.0
DS-R1-Distill-Qwen-7B-ScaleQuest	**93.0**	**91.0**	**53.3**	**90.0**	**81.8**

Table 8: Results of ScaleQuest-LongCoT.

CodeFeedback-Filtered using Qwen2.5-Coder-7B-Instruct, and the results indicate that our generated questions are of higher quality. This further demonstrates the effectiveness of the ScaleQuest method.

D Insights behind model selection

In our works, we use multiple models, e.g., DSMath-7B-RL, Qwen2-Math-7B-Ins, GPT-4o-mini, and DSMath-7B-Base, which may cause confusion for model selection. In response, we also supplement our approach with a simpler setup. We use Qwen2-Math-7B-Ins for training question generators, constructing optimization data for QPO, and performing solvability & difficulty filtering, as well as for response generation. For reward filtering, InternLM-7B-Reward remained unchanged. The results, as shown in Figure 11 (ScaleQuest-Simple result), indicate that our approach continues to demonstrate superior performance compared to existing datasets. Additionally, we summarize these insights on model selection for domain adaptation:

- **Selection of base model for training question generator:** The self-synthesis generation paradigm heavily relies on the inherent knowledge of the problem-solving model itself (Xu et al., 2024). Therefore, a domain-specific model is essential. For example, Qwen2-Math-Ins is suitable for mathematical reasoning, while Qwen2.5-Coder-Ins fits well for code reasoning. Furthermore, using multiple question generators often leads to more diverse and higher-quality questions (as discussed in section 3.3).
- **Selection of model for constructing optimization data:** Well-aligned, general-purpose models, such as Llama3.1-70B and GPT-4o-mini, tend to perform better than domain-specific models, as illustrated in Figure 4.
- **Selection of Response Generation Model & Reward Model:** These can be selected based on their performance on the corresponding mathematical tasks.

We believe that the methodology and the experience in selecting models are always more critical than the chosen models themselves. With the continuous advancements in the open-source community, we are confident that stronger models will undoubtedly produce even better datasets when applying our approach.

E Additional Experiments

E.1 Evaluation Results on More Out-of-Domain (OOD) Benchmarks

In addition to College Math and Olympiad Bench, we include two additional OOD benchmarks: GSM-Hard (Gao et al., 2023) and MathChat (Liang et al., 2024). GSM-Hard is constructed by modifying the questions in GSM8K, replacing the numbers with larger, less common ones. From MathChat, we select two problem-solving tasks: follow-up QA and error correction. The results are summarized in Table 10. In more fine-grained OOD evaluations, our model continues to perform on par with Qwen2-Math-7B-Ins, further demonstrating

Model	# Samples (K)	HumanEval	MBPP	BigCodeBench	Average
Qwen2.5-Coder-CFB	156	79.3	77.2	35.6	64.0
Qwen2.5-Coder-CFB-Aug	156	84.1	**84.7**	39.0	69.3
Qwen2.5-Coder-ScaleQuest	156	**86.6**	83.1	**40.0**	**69.9**

Table 9: Results of ScaleQuest in Code Reasoning Task. All results are based on Qwen2.5-Coder-7B-Base. CFB refers to the CodeFeedBack-Filtered Dataset. We augment the responses for the problems in CodeFeedback-Filtered using Qwen2.5-Coder-7B-Instruct with reward filtering, creating a new dataset referred to as CFB-Aug.

our ScaleQuest Model's generalization capability and highlighting the generated data's robustness.

E.2 Comparison Under Equal Training Data Volume

In the right panel of Figure 1, we plot the scaling trends of model performance with increasing data volume, showcasing the superiority of the ScaleQuest method when using the same amount of data. To further ensure a fair comparison, we randomly sample the same number of training examples from open-source datasets for training. Specifically, we sample 400K examples from MetaMath, DART-Math, NuminaMath, and our dataset (for MetaMath, which contains 395K examples in total, all samples are used). The results are presented in Table 11. We observe that with the same amount of training data, our dataset demonstrates significantly higher instruction tuning effectiveness compared to other datasets.

E.3 Training Data Volume on QPO

QPO is designed to enhance the solvability and difficulty of the question generator. We investigate the impact of training data volume by using GPT-4o-mini as the optimization model. The training data volume is controlled at 5K, 10K, 15K, 20K, and 40K, with Qwen2-Math-7B-QFT serving as the base model. We evaluate the performance of the trained question generator in terms of solvability and difficulty. The results are shown in Figure 9. As the amount of training data increases, both the solvable rate and difficulty of the questions generated by the question generator improve, gradually converging around 20K training examples. We believe that maintaining the training data at approximately 10K represents a more suitable balance between training cost and model performance.

E.4 Additional Results Based on Different Base Models

We have supplemented Table 2 with the results for the other three base models, as shown in Table 12. Under the same response generation process, our approach consistently outperforms existing datasets across all four base models, further demonstrating the superiority of our method.

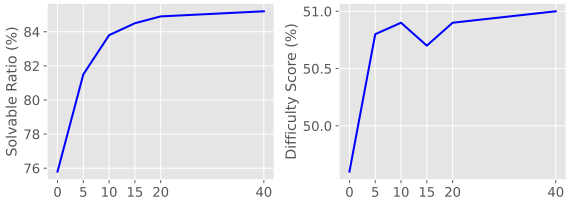

Figure 9: Performance of QPO in different training data volume. The evaluation covers the solvable ratio and difficulty score, following the same evaluation procedure as in Figure 5.

F Prompt Template

Model	GSM-Hard	Follow-up QA			Error Correction	Average
		R1	R2	R3		
Qwen2-Math-7B-Instruct	68.3	89.5	62.4	53.5	89.9	72.7
Qwen2-Math-7B-ScaleQuest	66.3	89.7	61.7	53.5	91.1	72.5

Table 10: The comparison between Qwen2-Math-7B-Ins and the ScaleQuest Model on GSM-Hard and MathChat. We choose Follow-up QA and Error Correction from MathChat for evaluation in problem-solving. R1, R2, and R3 represent different rounds in Follow-up QA.

Model	# Samples (K)	GSM8K	MATH	College Math	Olympiad Bench	Average
Qwen2-Math-7B-MetaMath	395	84.3	48.6	40.5	15.6	47.3
Qwen2-Math-7B-DART-Math	400	88.6	58.2	45.2	22.8	53.7
Qwen2-Math-7B-NuminaMath	400	82.0	65.8	44.9	29.2	55.5
Qwen2-Math-7B-ScaleQuest	400	**90.6**	**71.6**	**50.2**	**36.2**	**62.1**
Qwen2-Math-7B-ScaleQuest-Simple	400	89.4	69.9	48.8	33.6	60.4

Table 11: Results on four mathematical reasoning benchmarks. All results are based on Qwen2-Math-7B-Base. ScaleQuest-Simple is a simplified version that only utilizes Qwen2-Math-7B-Ins for QFT, QPO, and question filtering, and InternLM-7B-Reward for reward filtering.

Model	GSM8K	MATH	College Math	Olympiad Bench	**Average**
Mistral-7B-MetaMath-Aug	77.0	34.1	18.6	8.6	34.6
Mistral-7B-OrcaMath-Aug	84.4	31.6	20.9	8.2	36.3
Mistral-7B-NumiMath-Aug	79.5	62.8	40.4	**30.4**	53.3
Mistral-7B-ScaleQuest	**88.5**	**62.9**	**43.5**	28.8	**55.9**
Llama3-8B-MetaMath-Aug	77.6	33.1	20.6	9.2	35.1
Llama3-8B-OrcaMath-Aug	83.2	32.6	19.4	8.6	36.0
Llama3-8B-NumiMath-Aug	79.1	62.9	39.3	**25.4**	51.7
Llama3-8B-ScaleQuest	**87.9**	**64.4**	**42.8**	25.3	**55.1**
Qwen2-Math-7B-MetaMath-Aug	88.5	68.5	47.1	33.0	59.3
Qwen2-Math-7B-OrcaMath-Aug	89.3	68.3	46.6	31.9	59.0
Qwen2-Math-7B-NumiMath-Aug	89.5	72.6	49.5	36.3	62.0
Qwen2-Math-7B-ScaleQuest	**89.7**	**73.4**	**50.0**	**38.5**	**62.9**

Table 12: Additional results of Table 2 on the other base models. All responses are generated using Qwen2-Math-7B-Instruct with the same reward filtering process. For baseline datasets, "-Aug" indicates that the responses have been enhanced.

Prompts for Problem Solvability Optimization

```
Please act as a professional math teacher.
Your goal is to create high quality math word problems to help students learn
math.
You will be given a math question.  Please optimize the Given Question and
follow the instructions.
To achieve the goal, please follow the steps:
# Please check that the given question is a math question and write detailed
solution to the Given Question.
# Based on the problem-solving process, double check the question is solvable.
# If you feel that the given question is not a meaningful math question, rewrite
one that makes sense to you. Otherwise, modify the Given question according to
your checking comment to ensure it is solvable and of high quality.
# If the question can be solved with just a few simple thinking processes, you
can rewrite it to explicitly request multiple-step reasoning.

You have five principles to do this:
# Ensure the optimized question only asks for one thing, be reasonable and
solvable, be based on the Given Question (if possible), and can be answered
with only a number (float or integer). For example, DO NOT ask, 'what is the
amount of A, B and C?'.
# Ensure the optimized question is in line with common sense of life.  For
example, the amount someone has or pays must be a positive number, and the
number of people must be an integer.
# Ensure your student can answer the optimized question without the given
question.  If you want to use some numbers, conditions or background in the
given question, please restate them to ensure no information is omitted in your
optimized question.
# Please DO NOT include solution in your question.

Given Question: problem
Your output should be in the following format:
CREATED QUESTION: [your created question]
VERIFICATION AND MODIFICATION: [solve the question step-by-step and modify it
to follow all principles]
FINAL QUESTION: [your final created question]
```

Figure 10: Prompts used to optimize the solvability of questions for QPO Training.

Prompts for Problem Difficulty Optimization

You are an Math Problem Rewriter that rewrites the given #Problem# into a more complex version.
Please follow the steps below to rewrite the given "#Problem#" into a more complex version.

Step 1: Please read the "#Problem#" carefully and list all the possible methods to make this problem more complex (to make it a bit harder for well-known AI assistants such as ChatGPT and GPT4 to handle). Note that the problem itself might be erroneous, and you need to first correct the errors within it.
Step 2: Please create a comprehensive plan based on the #Methods List# generated in Step 1 to make the #Problem# more complex. The plan should include several methods from the #Methods List#.
Step 3: Please execute the plan step by step and provide the #Rewritten Problem#. #Rewritten Problem# can only add 10 to 20 words into the "#Problem#".
Step 4: Please carefully review the #Rewritten Problem# and identify any unreasonable parts. Ensure that the #Rewritten Problem# is only a more complex version of the #Problem#. Just provide the #Finally Rewritten Problem# without any explanation and step-by-step reasoning guidance.

Please reply strictly in the following format:
Step 1 #Methods List#:
Step 2 #Plan#:
Step 3 #Rewritten Problem#:
Step 4 #Finally Rewritten Problem#:

#Problem#: Problem

Figure 11: Prompts used to optimize the difficulty of questions for QPO Training.

Prompts for Problem Solvability Check

Please act as a professional math teacher.
Your goal is to determine if the given problem is a valuable math problem. You need to consider two aspects:
1. The given problem is a math problem.
2. The given math problem can be solved based on the conditions provided in the problem (You can first try to solve it and then judge its solvability).

Please reason step by step and conclude with either 'Yes' or 'No'.

Given Problem: Problem

Figure 12: Prompts used to check the solvability of questions.

Prompts for Difficulty Classification

```
# Instruction

You first need to identify the given user intent and then label the
difficulty level of the user query based on the content of the user query.

## User Query
Input

## Output Format
Given the user query, in your output, you first need to identify the user
intent and the knowledge needed to solve the task in the user query.
Then, rate the difficulty level of the user query as very easy, easy, medium,
hard, or very hard.

Now, please output the user intent and difficulty level below in a json
format by filling in the placeholders in []:
{{
“intent”: “The user wants to [....]”,
“knowledge”: “To solve this problem, the models need to know [....]”,
“difficulty”: “[very easy/easy/medium/hard/very hard]”
}}
```

Figure 13: The prompts used to judge the difficulty level of questions.

Prompts for Topic Classification

As a mathematics education specialist, please analyze the topics of the provided question and its answer.
Specific requirements are as follows:
1. You should identify and categorize the main mathematical topics involved in the problem. If knowledge from non-mathematical fields is used, it is classified into Others - xxx, such as Others - Problem Context.
2. You should put your final answer between <TOPIC> and </TOPIC>.
—
Question: Compute $\cos 330^\circ$.

Answer: We know that $330^\circ = 360^\circ - 30^\circ$.
Since $\cos(360^\circ - \theta) = \cos\theta$ for all angles θ,
we have $\cos 330^\circ = \cos 30^\circ$.
Since $\cos 30^\circ = \frac{\sqrt{3}}{2}$,
we can conclude that $\cos 330^\circ = \boxed{\frac{\sqrt{3}}{2}}$.

Analysis: <TOPIC>Trigonometry - Cosine Function</TOPIC>
—
Question: Question

Answer: Answer

Analysis:

Figure 14: The prompts used for topic classification.

Examples for Solvability Optimization

Problems 1 (Before Optimization):
There are 10 survivors in an emergency room. Each survivor is either a child, a woman, or a man. If there are 4 men and 3 times as many women as men, how many children are there?
Problems 1 (After Optimization):
There are 10 survivors in an emergency room. Each survivor is either a child, a woman, or a man. If there are 4 men and an equal number of women as men, how many children are there?

Problems 2 (Before Optimization):
How many sides does a polygon have if it is a regular polygon?
Problems 2 (After Optimization):
How many sides does a regular polygon have if each interior angle is 120 degrees?

Problems 3 (Before Optimization):
Find the sum of the first three terms of this series.
Problems 3 (After Optimization):
Calculate the sum of the first three terms of the arithmetic series where the first term is 5 and the common difference is 3.

Figure 15: Three examples for solvability optimization by GPT-4o-mini.

Examples for Difficulty Optimization

Problems 1 (Before Optimization):
How many 4-digit positive integers are there?
Problems 1 (After Optimization):
How many 4-digit positive integers can be formed using non-repeating digits where the sum of these digits must be even, and the integers fall within the range of 1000 to 9999?

Problems 2 (Before Optimization):
The average of 15 numbers is 32. An additional number is then added to the list, and the new average of the 16 numbers is 34. What number was added to the list?
Problems 2 (After Optimization):
The average of 15 positive integers is 32, but one integer fluctuates to 30 before adding a new number. After adding this new number, the average of the 16 integers becomes 34. Calculate the added number and find the standard deviation of all 16 integers, considering their ascending order.

Problems 3 (Before Optimization):
A fair coin is tossed 50 times, what is the probability of getting heads at least 25 times?
Problems 3 (After Optimization):
A fair coin is tossed 50 times; what is the probability of obtaining heads at least 25 times, and can you also calculate the expected number, variance, and standard deviation of heads while determining the likelihood that the total number of heads exceeds 30?

Figure 16: Three examples for difficulty optimization by GPT-4o-mini.

ScaleQuest Data Example 1

QUESTION:
My orthogonal projection matrix P projects onto the plane with equation

$$x + y + z = 0.$$

Find the matrix P.

SOLUTION:
To find the orthogonal projection matrix P that projects onto the plane with the equation $x + y + z = 0$, we can follow these steps:
1. **Identify the normal vector to the plane:**
The plane $x + y + z = 0$ has a normal vector $\mathbf{n} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$.
2. **Express the projection matrix in terms of the normal vector:**
The orthogonal projection matrix P onto the plane can be written as:

$$P = I - \frac{\mathbf{n}\mathbf{n}^T}{\mathbf{n}^T\mathbf{n}}$$

where I is the 3×3 identity matrix, $\mathbf{n}$ is the normal vector, and $\mathbf{n}^T$ is the transpose of the normal vector.
3. **Calculate $\mathbf{n}^T\mathbf{n}$:**

$$\mathbf{n}^T\mathbf{n} = \begin{pmatrix} 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} = 1^2 + 1^2 + 1^2 = 3$$

4. **Calculate $\mathbf{n}\mathbf{n}^T$:**

$$\mathbf{n}\mathbf{n}^T = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

5. **Substitute these into the projection matrix formula:**

$$P = I - \frac{1}{3} \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

where I is the 3×3 identity matrix:

$$I = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

6. **Perform the matrix subtraction:**

$$P = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} - \frac{1}{3} \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

$$P = \begin{pmatrix} \frac{2}{3} & -\frac{1}{3} & -\frac{1}{3} \\ -\frac{1}{3} & \frac{2}{3} & -\frac{1}{3} \\ -\frac{1}{3} & -\frac{1}{3} & \frac{2}{3} \end{pmatrix}$$

Therefore, the orthogonal projection matrix P is:

$$\boxed{\begin{pmatrix} \frac{2}{3} & -\frac{1}{3} & -\frac{1}{3} \\ -\frac{1}{3} & \frac{2}{3} & -\frac{1}{3} \\ -\frac{1}{3} & -\frac{1}{3} & \frac{2}{3} \end{pmatrix}}$$

Figure 17: An example from our constructed dataset.

ScaleQuest Data Example 2

QUESTION:
Suppose a, b are positive real numbers such that $a + b = 1$.
(a) minimize $\frac{1}{a} + \frac{1}{b}$;
(b) minimize $a^2 + b^2$.

SOLUTION:
To solve these problems, we will use the method of Lagrange multipliers and the AM-HM inequality.
Part (a): Minimize $\frac{1}{a} + \frac{1}{b}$ given $a + b = 1$
First, we use the method of Lagrange multipliers. Define the function $f(a, b) = \frac{1}{a} + \frac{1}{b}$ and the constraint $g(a, b) = a + b - 1 = 0$.
The Lagrangian is:

$$\mathcal{L}(a, b, \lambda) = \frac{1}{a} + \frac{1}{b} + \lambda(a + b - 1)$$

Taking the partial derivatives and setting them to zero, we get:

$$\frac{\partial \mathcal{L}}{\partial a} = -\frac{1}{a^2} + \lambda = 0 \implies \lambda = \frac{1}{a^2}$$

$$\frac{\partial \mathcal{L}}{\partial b} = -\frac{1}{b^2} + \lambda = 0 \implies \lambda = \frac{1}{b^2}$$

$$\frac{\partial \mathcal{L}}{\partial \lambda} = a + b - 1 = 0$$

Since $\lambda = \frac{1}{a^2}$ and $\lambda = \frac{1}{b^2}$, we have:

$$\frac{1}{a^2} = \frac{1}{b^2} \implies a^2 = b^2 \implies a = b \quad (\text{since } a, b > 0)$$

Substituting $a = b$ into the constraint $a + b = 1$, we get:

$$a + a = 1 \implies 2a = 1 \implies a = \frac{1}{2} \implies b = \frac{1}{2}$$

Now, we calculate $\frac{1}{a} + \frac{1}{b}$:

$$\frac{1}{a} + \frac{1}{b} = \frac{1}{\frac{1}{2}} + \frac{1}{\frac{1}{2}} = 2 + 2 = 4$$

Thus, the minimum value of $\frac{1}{a} + \frac{1}{b}$ is $\boxed{4}$.
Part (b): Minimize $a^2 + b^2$ given $a + b = 1$
We use the identity for the square of a sum:

$$(a + b)^2 = a^2 + b^2 + 2ab$$

Given $a + b = 1$, we have:

$$1^2 = a^2 + b^2 + 2ab \implies 1 = a^2 + b^2 + 2ab \implies a^2 + b^2 = 1 - 2ab$$

To minimize $a^2 + b^2$, we need to maximize ab. Using the AM-GM inequality:

$$\frac{a + b}{2} \geq \sqrt{ab} \implies \frac{1}{2} \geq \sqrt{ab} \implies \left(\frac{1}{2}\right)^2 \geq ab \implies \frac{1}{4} \geq ab$$

The maximum value of ab is $\frac{1}{4}$, which occurs when $a = b = \frac{1}{2}$.
Substituting $ab = \frac{1}{4}$ into the expression for $a^2 + b^2$:

$$a^2 + b^2 = 1 - 2ab = 1 - 2\left(\frac{1}{4}\right) = 1 - \frac{1}{2} = \frac{1}{2}$$

Thus, the minimum value of $a^2 + b^2$ is $\boxed{\frac{1}{2}}$.

Figure 18: An example from our constructed dataset.

ScaleQuest Data Example 3

QUESTION:
Compute

$$\sum_{n=1}^{\infty} \frac{4^n + 5^n + 6^n}{12^n}.$$

SOLUTION:
To compute the sum $\sum_{n=1}^{\infty} \frac{4^n+5^n+6^n}{12^n}$, we can split the sum into three separate sums:

$$\sum_{n=1}^{\infty} \frac{4^n + 5^n + 6^n}{12^n} = \sum_{n=1}^{\infty} \frac{4^n}{12^n} + \sum_{n=1}^{\infty} \frac{5^n}{12^n} + \sum_{n=1}^{\infty} \frac{6^n}{12^n}.$$

Each of these sums is a geometric series. Let's evaluate each one separately.
1. For the first sum $\sum_{n=1}^{\infty} \frac{4^n}{12^n}$:

$$\sum_{n=1}^{\infty} \frac{4^n}{12^n} = \sum_{n=1}^{\infty} \left(\frac{4}{12}\right)^n = \sum_{n=1}^{\infty} \left(\frac{1}{3}\right)^n.$$

The sum of an infinite geometric series $\sum_{n=1}^{\infty} ar^{n-1}$ is given by $\frac{a}{1-r}$ for $|r| < 1$. Here, $a = \frac{1}{3}$ and $r = \frac{1}{3}$, so:

$$\sum_{n=1}^{\infty} \left(\frac{1}{3}\right)^n = \frac{\frac{1}{3}}{1 - \frac{1}{3}} = \frac{\frac{1}{3}}{\frac{2}{3}} = \frac{1}{2}.$$

2. For the second sum $\sum_{n=1}^{\infty} \frac{5^n}{12^n}$:

$$\sum_{n=1}^{\infty} \frac{5^n}{12^n} = \sum_{n=1}^{\infty} \left(\frac{5}{12}\right)^n.$$

Here, $a = \frac{5}{12}$ and $r = \frac{5}{12}$, so:

$$\sum_{n=1}^{\infty} \left(\frac{5}{12}\right)^n = \frac{\frac{5}{12}}{1 - \frac{5}{12}} = \frac{\frac{5}{12}}{\frac{7}{12}} = \frac{5}{7}.$$

3. For the third sum $\sum_{n=1}^{\infty} \frac{6^n}{12^n}$:

$$\sum_{n=1}^{\infty} \frac{6^n}{12^n} = \sum_{n=1}^{\infty} \left(\frac{6}{12}\right)^n = \sum_{n=1}^{\infty} \left(\frac{1}{2}\right)^n.$$

Here, $a = \frac{1}{2}$ and $r = \frac{1}{2}$, so:

$$\sum_{n=1}^{\infty} \left(\frac{1}{2}\right)^n = \frac{\frac{1}{2}}{1 - \frac{1}{2}} = \frac{\frac{1}{2}}{\frac{1}{2}} = 1.$$

Adding these three results together, we get:

$$\sum_{n=1}^{\infty} \frac{4^n + 5^n + 6^n}{12^n} = \frac{1}{2} + \frac{5}{7} + 1.$$

To add these fractions, we need a common denominator. The least common multiple of 2, 7, and 1 is 14. So we convert each fraction:

$$\frac{1}{2} = \frac{7}{14}, \quad \frac{5}{7} = \frac{10}{14}, \quad 1 = \frac{14}{14}.$$

Adding these fractions together, we get:

$$\frac{7}{14} + \frac{10}{14} + \frac{14}{14} = \frac{7 + 10 + 14}{14} = \frac{31}{14}.$$

Thus, the sum is:

$$\boxed{\frac{31}{14}}.$$

Figure 19: An example from our constructed dataset.