A Novel Instruction Tuning Method for Vietnamese Mathematical Reasoning using Trainable Open-Source Large Language Models

Quang-Vinh Nguyen[1]†, Thanh-Do Nguyen[1]†, Van-Vinh Nguyen[2], Khac-Hoai Nam Bui[1]*
[1]Viettel AI, Viettel Group, Vietnam
[2]Vietnam National University of Hanoi, Hanoi, Vietnam
{vinhnq29, dont15}@viettel.com.vn , vinhvn@vnu.edu.vn, nambkh@viettel.com.vn

Abstract

This study introduces **Si**mple **R**easoning with **C**ode (**SiRC**), a novel instruction fine-tuning method for solving mathematical reasoning problems, particularly designed for Vietnamese, which is considered a low-resource language. Specifically, solving mathematical problems requires strategic and logical reasoning, which remains challenging in this research area. This paper presents a simple yet effective instruction fine-tuning method for mathematical reasoning. Unlike previous approaches, our proposed method effectively combines chain-of-thought reasoning with code generation without requiring a sophisticated inference procedure. Furthermore, we focus on exploiting small open-source large language models (LLMs) for the Vietnamese language. In this regard, we first introduce a trainable Vietnamese mathematical reasoning dataset, which is named **`ViMath-InstructCode`**. The proposed dataset is then used for fine-tuning open-source LLMs (e.g., less than 10 billion parameters). Experiments conducted on our custom **`ViMath-Bench`** dataset, the largest benchmarking dataset focusing on Vietnamese mathematical problems, indicate the promising results of our proposed method. Our source code and dataset are available for further exploitation[1].

1 Introduction

Large language models (LLMs), including closed sources (e.g., GPT series (OpenAI, 2023)) and open sources (e.g., Llama series (Touvron et al., 2023)) have become fundamental in advancing natural language processing (NLP). These models achieve remarkable language comprehension and generation abilities, which advances many applications in text generation, code assistance, and mathematical reasoning. Notably, leading proprietary LLMs like GPT-4 and Claude excel in mathematical tasks, as evidenced by their top rankings on benchmarks such as GSM8K and MATH. However, smaller open-source models (fewer than 10 billion parameters) significantly lag in performance. It is challenging for open source to achieve similar capabilities due to the nature of mathematical problem-solving, which requires precise multiple reasoning steps, symbolic manipulation, and complex computation (Ahn et al., 2024).

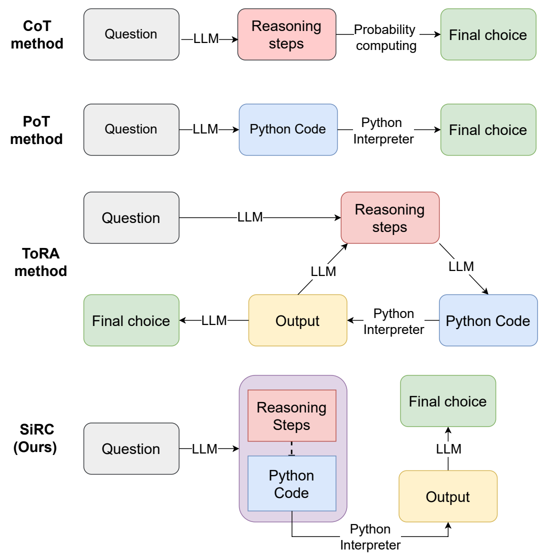

Figure 1: Comparative analysis of previous works with our approach for mathematical reasoning task using LLM with finetune instruction: Traditional method using reasoning step (CoT method); PoT uses Codex to generate text to programming language statements; ToRA employing multiple LLM calls within an LLM agent setting; Our proposed method uses LLM to generate both reasoning step and code generation within a single call LLM.

Technically, a potential solution is to special-

[1]https://github.com/quangvinh2110/vietnamese-math-reasoning
† Equal contribution
* Corresponding author

ize general-purpose LLMs in mathematics via supervised fine-tuning by distilling the knowledge from larger teacher models into smaller student models (Fu et al., 2023; Liang et al., 2023). An early approach uses chain-of-thought (CoT) explanations of existing data or extra CoT-style data generated by the larger models to train the smaller student model (Liu et al., 2023). Sequentially, (Chen et al., 2022a) proposes Program of Thoughts (PoT), which uses Codex (Chen et al., 2021) to generate text-to-programming language statements to find the answer. A recent approach uses the emerging LLM agent concept (Xi et al., 2023) to combine the two aforementioned approaches to improve the performance of mathematical reasoning (Gou et al., 2023a).

Despite various attempts to narrow the gap between closed-source and open-source models, the most cost-effective method for solving mathematical problems remains unresolved. Naively applying strategies like using chain-of-thought (CoT) or code generation to solve problems has not produced optimal results. Furthermore, employing multiple LLM calls within an agent setting (Gou et al., 2023a) incurs higher costs. Additionally, research on solving mathematical problems in Vietnamese, a low-resource language, is still nascent due to a lack of studies in this area.

In this regard, this study proposes **SiRC**, a simple effective instruction finetuning approach by combining chain-of-thought reasoning with code generation. Conceptual comparisons among **SiRC** and other previous approaches in this research field are illustrated in Figure 1. Generally, our main contributions in this study are threefold as follows:

- We propose **SiRC** framework, a simple and novel approach for solving elementary-level mathematical problems using a mixture of chain-of-thought reasoning and code generation (Figure 1). Empirical studies have demonstrated that this approach is effective for Vietnamese mathematical problems at this level and outperforms the naive implementation of CoT and code transferring.

- We present the first large-scale Vietnamese elementary mathematical dataset of 8k samples collected from various trusted sources, which we called **`ViMath-Bench`**. Furthermore, we augment this dataset using strong teacher models (`Llama3-70B-Instruct`[2] and `Qwen2-72B-Instruct`[3]), resulting in **`ViMath-InstructCode`** dataset consisting of 14k training samples. We also explored other synthetic data construction approaches. To the best of our knowledge, this is the first comprehensive study of Vietnamese mathematical reasoning.

- We release a series of models finetuned with **`ViMath-InstructCode`** dataset, which yield superior performance on **`ViMath-Bench`** test set. We hope that these models will establish a solid baseline for future research in mathematical reasoning in Vietnamese.

2 Literature Review

Human-annotated Math Datasets: Solving math word problems using Large Language Models (LLMs) has attracted extensive research efforts to create diverse datasets that enhance the model's mathematical reasoning capabilities, particularly in the English language. While early large-scale datasets like Dolphin18K (Huang et al., 2016) provided a foundation, they lacked detailed information on deriving the final answer, limiting their usefulness in teaching mathematical reasoning to models. Similarly, the AQuA-RAT dataset (Ling et al., 2017) has quality issues, including over-templatization and incorrect solutions. More recent math datasets have been designed with a focus on including detailed explanations and a diverse range of natural language expressions to provide more useful signals during model training. Notable examples are MathQA (Amini et al., 2019), GSM8K (Cobbe et al., 2021), and MATH (Hendrycks et al., 2021), which have improved the quality of the data by including questions requiring multiple solving steps as well as providing correct solutions. However, the language barrier presents a challenge, as these datasets are primarily in English, making them less directly beneficial for low-resource language research. Especially in Vietnamese, as far as we know, there has not been any large-scale dataset dedicated to math word problems.

Synthetic Data Construction: Some LLMs exhibit advanced mathematical reasoning and

[2]`https://huggingface.co/meta-llama/Meta-Llama-3-70B-Instruct`

[3]`https://huggingface.co/Qwen/Qwen2-72B-Instruct`

tool use abilities, making the idea of distilling knowledge from these models to student model highly attractive. `nvidia/OpenMathInstruct-1`[4] extracts problems from GSM8K and MATH training subsets and synthetically generate solutions by using `Mixtral 8x7b`[5] model to use a mix of text reasoning and code blocks executed by Python interpreter. `Tiger-Lab/MathInstruct`[6] compiles a list of high-quality and diverse math instruction-tuning datasets augmented by GPT-4. Several other approaches use capable LLMs to augment existing math datasets, such as evolving the difficulties of the questions (Luo et al., 2023) or deriving detailed solution trajectories interleaving rationales and code (Gou et al., 2023b).

Mathematical Reasoning and Tool Integration: The chain-of-thought (Wei et al., 2022) prompting technique, which instructs a model to divide a problem into smaller, manageable sub-problems, enhances reasoning tasks significantly (referenced in CoT and least-to-most papers). This method has its merits in mathematical reasoning as well (as seen in wizard math studies), but its effectiveness diminishes when tasks require symbolic manipulation and computations. An alternative strategy involves training models to create code that solves problems, then utilizing computational tools like a Python interpreter to execute the code (Chen et al., 2022b). However, relying solely on code generation is not effective for theoretical questions or in scenarios with complex natural language, as it may lack sufficient rationale. (Gou et al., 2023b) combines chain-of-thought with code generation to improve performance, though it requires multiple interactions with large language models (LLM). All mentioned approaches are actively explored with English datasets in focus, however there has been no study on this subject in Vietnamese.

3 ViMath-Bench Dataset

In this section, we present the construction procedure of the Vietnamese mathematical reasoning dataset, **ViMath-Bench**. To the best of our knowledge, this is the first dataset created for Vietnamese mathematical reasoning. The pipeline of the data construction is illustrated in the Figure 2, which are sequentially described as follows:

3.1 Data Sources

Our dataset is derived from three prominent Vietnamese online educational platforms: Tailieumoi[7], Hamchoi[8], and Vietjack[9]. These websites serve as comprehensive resources for general education in Vietnam, catering to a diverse audience including students, teachers, and parents. They provide solutions to textbook and workbook problems, reference materials for grades 1 to 12 across various subjects, and lesson plans for teachers. Notably, all content on these websites is freely accessible. For our study, we specifically targeted multiple-choice questions (MCQs) from grades 3 to 5, collecting approximately 20,000 questions. The Vietjack website provided fields such as `question`, `choices`, `full answer`, and `right choice`. However, Tailieumoi and Hamchoi did not offer the `right choice` field, necessitating additional steps to complete the dataset.

3.2 Preprocessing

To ensure the quality of the data, we implemented a multi-step preprocessing pipeline. First, we normalize the data to follow a consistent and standardized representation:

- **Text Normalization**: All text data was normalized to the NFC standard to ensure consistent character encoding. Vietnamese tones were also standardized to maintain uniformity across the dataset.

- **Format Conversion**: We converted HTML formats to Markdown using the Pandoc[10] library. This conversion not only saved tokens during the model training and inference process but also facilitated easier processing and analysis.

Subsequently, we implemented a rigorous filtering process to ensure the data was of the highest quality:

- **Exclusion of Non-Relevant Samples:** We filtered out samples containing tables and images, as our current focus does not include multimodal data.

[4] `https://huggingface.co/datasets/nvidia/OpenMathInstruct-1`

[5] `https://huggingface.co/mistralai/Mixtral-8x7B-v0.1`

[6] `https://huggingface.co/datasets/TIGER-Lab/MathInstruct`

[7] `https://tailieumoi.vn/`

[8] `https://hamchoi.vn/`

[9] `https://www.vietjack.com/`

[10] `https://pandoc.org/`

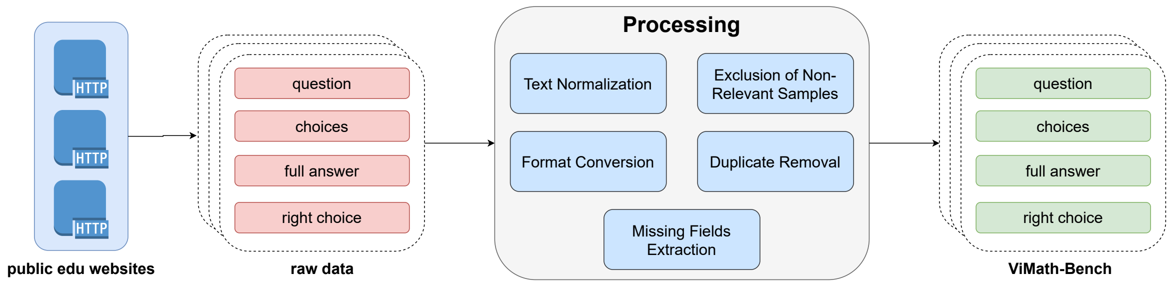

Figure 2: ViMath-Bench Dataset.

- **Duplicate Removal:** We employed an edit distance metric to identify and remove duplicate questions. Specifically, if two samples had an edit distance of greater than 90, one sample was discarded. Preference was given to retaining samples due to the completeness of its data fields.

- **Missing Fields Extraction:** For the remaining data from the available resources, we employed a set of rules to extract the `correct choice` field for `full answer`. This method allowed us to successfully retrieve the correct answers for approximately 70% of the samples. The remaining 30% of the data, which lacked this critical field, were subsequently removed.

After these preprocessing steps, the dataset contains 8.4K samples, which we name **ViMath-Bench**. Each sample of the dataset contains four fields: `question`, `choices`, `correct_choice`, and `full_answer`, which are beneficial for both training and evaluation. This dataset is then divided into 7.1K training samples and 1.3K test samples to facilitate the evaluation of our models.

4 Methodology

We introduce **Simple Reasoning with Code (SiRC)** framework, which is a simplified approach to solving elementary mathematical problems using both CoT reasoning and code generation. We leverage a teacher-student framework to distill knowledge from larger open-source LLMs to smaller, more resource-efficient models, tailored specifically for Vietnamese mathematical reasoning tasks.

4.1 SiRC Inference Procedure

We propose a novel, efficient approach to address the arithmetic and calculation challenges faced by large language models (LLMs), described in Algorithm 1. Our method integrates Python code generation into the problem-solving process in a unique way. Unlike previous studies that either generated only code all at once or used iterative reasoning and code generation (which can be cost-inefficient because of multiple LLM calls), our approach simplifies the process by combining reasoning and coding in a single, structured LLM response:

- **Step-by-Step Reasoning:** The LLM first outlines all necessary steps to solve the math problem without performing any calculations.

- **Python Code:** Immediately following the reasoning, the LLM generates Python code to execute the required calculations.

This seemingly two-step process is completed in a single LLM call, which not only simplifies the reasoning-code generation in a chained multi-agent setup but also enhances the effectiveness of the solution. By separating the logical problem-solving steps from the actual computation, our method provides a clear, executable pathway to solve complex mathematical problems. After finishing the reasoning-code generation and code execution, **SiRC** makes the last LLM call to generate the final answer to the question.

4.2 Teacher-Student Framework

To implement our **SiRC** approach effectively, we develop **`ViMath-InstructCode`**, a synthetic dataset designed to enable trainable open-source LLMs to adopt our proposed method. We employ a knowledge distillation approach (Semnani et al., 2023), utilizing larger models as teachers to transfer knowledge to smaller, more resource-efficient student models. In the most general case, this framework allows multiple strong larger models to act as teachers, enabling the available smaller models to learn from all of them and obtain combined capabilities. This process allows us to create

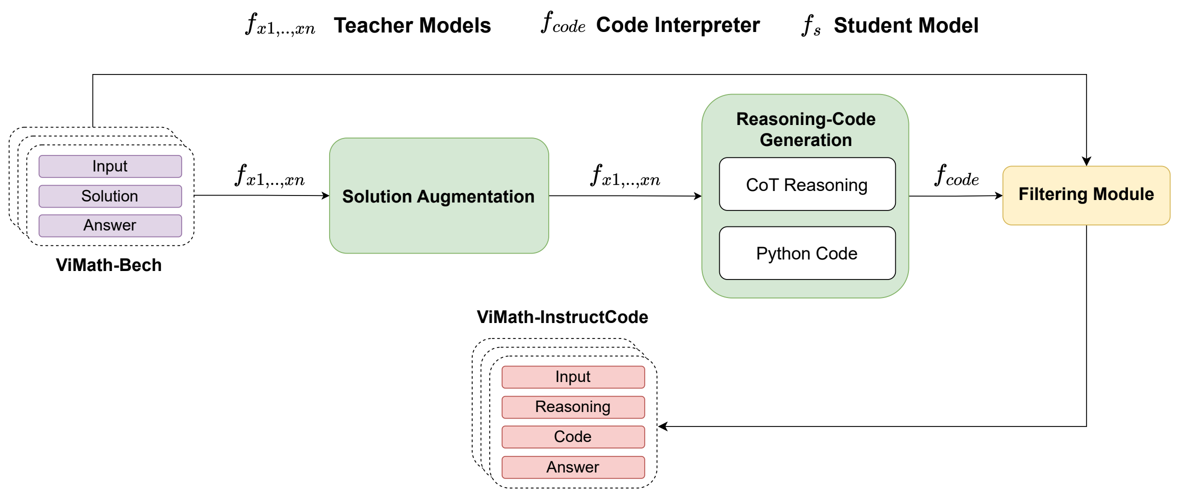

Figure 3: Overview of ViMath-Instruct-Code dataset construction

Algorithm 1 SiRC Inference Procedure

```
Require: Multiple-choice question of problem $P$
Ensure: Solution to $P$
1: Reasoning-Code Generation (LLM call): Generate all necessary steps to solve $P$ as textual guidance only + Python code to execute the required calculations based on the outlined steps.
2: Execution (Python interpreter): Extract the generated Python code and execute to obtain the solution to $P$.
3: Answer Generation (LLM call): Generate final choice for $P$ based on the output of previous steps.
```

more compact versions of the LLMs that are tailored for Vietnamese mathematical reasoning tasks while maintaining the ability to perform both step-by-step reasoning and code generation as outlined in our SiRC framework.

4.3 Construction of `ViMath-InstructCode` Dataset

Following the Teacher-Student framework, we specifically use `Llama-3-70B` and `Qwen-2-72B` as the teacher models, with publicly available small models serving as the distilled versions. We selected these two models because they usually demonstrate different reasoning responses to the same math problem, making it advantageous to learn from both. However, due to resource limitations, we could not extend our experiments to other large models or closed-source LLM APIs. Specifically, **`ViMath-InstructCode`** is constructed based on the **`ViMath-Bench`** training set, which is sequentially illustrated as follows (Figure 3)):

- **Input Data:** The input data consists of math problems collected and preprocessed as detailed in section 3. This dataset, referred to as **`ViMath-Bench`**, serves as the foundation for generating step-by-step solutions. Specifically, we extracted the training split and augmented it in subsequent steps to ensure it can be efficiently trained by language models.

- **Solution Augmentation**: Upon reviewing the full answer fields of the crawled data, we observed that the solutions often have an undesirable format where the conclusion precedes the explanation. This format may be unintuitive for the model to learn from, as learning from data of this format would teach the model to predict the answer before reasoning about the question. Additionally, the solutions lack depth, as they do not provide step-by-step explanations, thereby failing to offer a rich signal in reasoning for the model to learn. To address these issues, we decided to augment the crawled solutions using the teacher models. These models enhance the solutions in two ways: firstly, by adding a detailed explanation that outlines every step and computation involved in solving the problem; and secondly, by formatting the augmented solution so that the explanation precedes the final answer. At the end of this step, we obtain a dataset of pure Chain-of-Thought fashion.

- **Reasoning-Code Generation:** In this step, we further augment the detailed step-by-step solutions from the previous step to obtain data with the desired structure. This structure contains reasoning followed by code, which are central to our **SiRC** framework. Specifically, we prompted the teacher models to first reformat the solutions from the previous step by eliminating all computations and preserving only the reasoning steps. After completing the reformatting, the models then continue to generate Python code to solve problems that require calculations. The input of this process is a *detailed solution* (with chain-of-thought fashion which details on both reasoning and computation), and the output is *textual guidance* (without computation) and *Python code* (that executes necessary computations).

- **Data Filtering:** The generated data from the previous step are passed through a filtering module. This module extracts and executes the Python code, then compares the extracted answer from the code's output with the provided correct answer to verify its correctness. Samples with incorrect output are discarded. Additionally, edit distance is used to deduplicate the generated solutions and code snippets.

5 Experimental Setup

5.1 Evaluation Metrics

The primary metric for evaluating the performance of the models was accuracy. This metric measures the percentage of questions q where the model gives the correct final answer. The accuracy is calculated as follows:

$$Accuracy(Q) = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \{\texttt{SiRC}(q_i, C_i) == a_i\}$$

where q_i is the i-th question in a set of questions Q, C_i is the set of corresponding choices for question q_i. $\texttt{SiRC}(q_i, C_i)$ is the final answer concluded by running the **SiRC** inference procedure on q_i, which is compared with the correct choice a_i of q_i. Accuracy provides a straightforward and intuitive measure of the model's effectiveness in solving mathematical reasoning tasks.

5.2 Training datasets

We conducted extensive experiments using multiple training datasets, each representing a different approach to solving math problems with large language models (LLMs). These approaches include fine-tuning with an unprocessed crawled dataset as a baseline, Chain-of-Thought (CoT) reasoning, Programming-of-Thought (PoT), and our proposed **SiRC** framework. This diverse collection of training datasets allows for a comprehensive comparison, showcasing the effectiveness of our **SiRC** framework across various methodologies. Table 1 provides detailed information on the construction of each dataset, their characteristics, and the number of samples included. Notably, while the **`ViMath-Bench`** dataset contains only 7k training samples, its derivatives—**`ViMath-Reasoning`**, **`ViMath-Code`**, and most importantly, **`ViMath-InstructCode`**—each contains 14k training samples. This increase is due to the use of two teacher models, `Llama-3-70B` and `Qwen2-72B`, in generating these datasets.

Dataset	**Description**	**#Num.**
`ViMath-Bench`	Crawled dataset, described in section 3	7K
`ViMath-Reasoning`	Detailed Step-by-step reasoning with textual computation, generated by teacher models	14K
`ViMath-Code`	Only using code to solve the problem, generated by using teacher models	14K
`ViMath-InstructCode`	Structured reasoning followed by code, described in section 4.3	14K

Table 1: Details of Training Datasets for Different Approaches

5.3 Implementation

Hyperparameter	**Values**
batch size	128
epoch	3
learning rate	2e-4
learning rate scheduler	cosine
weight decay	0
cutoff-len	2048
lora-r	64
lora-alpha	128
lora-dropout	0.05
lora-target-modules	all linear

Table 2: Hyperparameters for the model fine-tuning

Model	No finetuning		Baseline Finetuning				Finetuning w/ SiRC (ours)	
	w/o CoT	w/ CoT	ViMath-Bench	ViMath-Reasoning	ViMath-Code	ViMath-Reasoning +Code	ViMath-InstructCode	ViMath-Reasoning +InstructCode
WizardMath-7B-V1.1	46.01	53.44	52.25	78.97	-	-	-	-
MetaMath-Mistral-7B	51.46	52.41	52.96	77.00	-	-	-	-
vinallama-7b-chat	40.79	38.97	38.26	57.87	-	-	-	-
Vistral-7B-Chat	44.98	63.56	51.30	74.31	-	-	-	-
Llama-3-8B-Instruct	75.97	67.98	73.60	83.32	85.22	86.01	**86.4**	**87.27**
Qwen2-7B-Instruct	83.4	84.82	79.68	88.38	87.98	88.14	**90.83**	**91.15**
deepseek-math-7b-rl	89.49	89.57	82.53	88.38	88.93	**89.64**	89.49	**90.59**
Llama-3-70B-Instruct	89.80	90.28	-	-	-	-	-	-
Qwen2-72B-Instruct	92.09	92.09	-	-	-	-	-	-

Table 3: Performance comparison of models under different finetuning conditions. The two best results in each row are in **bold**.

Backbone Model: In our experiments, we use a diverse selection of open-source language models as backbones. Firstly, we choose models specifically trained for mathematical problem-solving: `WizardMath-7B-V1.1` and `MetaMath-Mistral-7B`. However, these models lack native support for Vietnamese. To address this, we then select models trained with a sufficient amount of Vietnamese data: `vinallama-7b-chat` and `vistral-7b-chat`, though these are not specifically designed for solving math problems. Finally, we include some of the latest multilingual models which also show strong coding and mathematical capabilities: `Qwen2-7B-Instruct`, `Llama-3-8B-Instruct`, and `deepseek-math-7b-rl`. In total, we utilize seven models as our backbones.

Hyperparameters: For fine-tuning the backbones, we employed the Low-Rank Adaptation (LoRA) technique across all models. To ensure a fair comparison, we kept the hyperparameters consistent for every model. Detailed information regarding these hyperparameters is provided in Table 2.

6 Results

Table 3 presents a detailed performance comparison of various models under different conditions. The effectiveness of the proposed **SiRC** framework is highlighted, demonstrating its impact on enhancing model accuracy in mathematical reasoning tasks.

6.1 Baselines

No Fine-Tuning Among the models evaluated without any fine-tuning, `deepseek-math-7b-rl` and `Qwen2-7B-Instruct` exhibited the highest accuracies without CoT prompting, achieving 89.49% and 83.4%, respectively. This underscores these models' robust baseline capabilities in mathematical reasoning when used out of the box.

CoT prompting had varied effects on model performance. For some models, such as `vistral-7b-chat` and `WizardMath-7B-V1.1`, it significantly boosted accuracy by nearly 19% and 7%, respectively. However, for other models, CoT prompting had little to no effect or even decreased performance. This indicates that the benefits of CoT prompting are not consistent across all models and may depend on specific model architectures or underlying training data.

Baseline Fine-Tuning When models were finetuned with the baseline **`ViMath`** datasets (as detailed in Table 1), there were notable improvements in performance compared with no finetuning setting. Augmented datasets, namely **`ViMath-Reasoning`** and **`ViMath-Code`** consistently outperform raw dataset **`ViMath-Bench`**, showing the clear advantage of using teacher models to generate synthetic training data. Notably, for all models, we experimented with finetuning by combining these two augmented datasets yielded the highest accuracies across all baseline configurations. However, we did not conduct training experiments with code-related data on the first four models, because they were not sufficiently trained with code data.

6.2 Main results

Finetuning with SiRC framework To enable models to follow the SiRC inference framework, we trained them on datasets that include our proposed **`ViMath-InstructCode`** dataset. Finetuning with only **`ViMath-InstructCode`** dataset enabled models to consistently surpass performance when trained with only **`ViMath-Reasoning`** or

Model	Qwen2-7B-Instruct	Llama-3-8B-Instruct	deepseek-math-7b-rl
w/o Solution Augmentation	89.80	83.64	**89.72**
w/o Filtering Module	87.83	85.69	89.49
use only `Llama3-70B-Instruct`	88.46	84.19	87.98
use only `Llama3-70B-Instruct` (sampled twice)	87.91	84.11	87.91
use only `Qwen2-72B-Instruct`	89.25	83.79	88.14
use only `Qwen2-72B-Instruct` (sampled twice)	90.75	84.58	89.17
full pipeline (ours)	**90.83**	**86.40**	89.49

Table 4: Ablation study results

`ViMath-Code`. Additionally, finetuning using the combined **`ViMath-{Reasoning+InstructCode}`** dataset yielded the highest accuracies observed, with `Qwen2-7B-Instruct` achieving an impressive 91.15%, outperforming all other configurations. This highlights the synergistic effect of integrating reasoning data with our proposed **`ViMath-InstructCode`** data, which collectively enhances model performance more effectively than using either dataset in isolation.

Finally, we ran inference on the teacher models, `Llama3-70B-Instruct` and `Qwen2-72B-Instruct`, achieving the highest no-finetuning inference accuracy of 92.09% with `Qwen2-72B-Instruct` when using CoT prompting. Although `Qwen2-72B-Instruct` outperformed `Qwen2-7B-Instruct` (finetuned with our **`ViMath-InstructCode`** and follow **SiRC** inference procedure), our model closely followed by only less than 1% accuracy. This again demonstrates the robustness of the teacher-student methodology and the effectiveness of our **SiRC** framework.

6.3 Ablation Study

To further understand the contribution of each component in our proposed **SiRC** framework and the **`ViMath-InstructCode`** dataset, we conducted an ablation study. This study helps to isolate and evaluate the impact of different components and steps in our dataset construction pipeline. The configurations tested and their corresponding results are summarized in Table 4.

The full pipeline, including solution augmentation, filtering steps, and utilizing two teachers (`Llama3-70B-Instruct` and `Qwen2-72B-Instruct`, each sampled once), achieved the highest performance for both `Qwen2-7B-Instruct` (90.83%) and `Llama-3-8B-Instruct` (86.40%), demonstrating the effectiveness of the complete process. Excluding the solution augmentation step, which provides detailed explanations and formatting by the teacher models, resulting in a performance drop across almost all models. Using only `Llama3-70B-Instruct` or `Qwen2-72B-Instruct` for generating data also led to lower performance, underscoring the need for diversity provided by multiple teacher models. Sampling twice with either teacher model showed negligible improvement, indicating that the added diversity from another teacher model is crucial. Excluding the filtering step resulted in decreased accuracy for `Qwen2-7B-Instruct` (87.83%) and `Llama-3-8B-Instruct` (85.69%), emphasizing the role of data quality control in enhancing model reliability and accuracy. Interestingly, `deepseek-math-7b-rl` maintains stable performance even without the filtering step, suggesting that this model may be more resilient to noise in the training data compared to others. Overall, these results demonstrate that each component of our **`ViMath-InstructCode`** construction pipeline, including solution augmentation, the use of multiple teacher models, and filtering, significantly contributes to the overall performance of the models, with the full pipeline consistently yielding the best results, confirming the robustness and effectiveness of our approach.

7 Conclusion

This study proposes a novel fine-tuning instruction approach for mathematical reasoning, which is specified for the Vietnamese language. Specifically, we present SiRC, an effective framework, which significantly enhances the mathematical reasoning capabilities of language models with minimal cost. Furthermore, by leveraging the **`ViMath-InstructCode`** dataset and combining it with reasoning datasets, the proposed framework achieves superior performance, underscoring the effectiveness of our approach. Accordingly, the experimental results indicate that diverse and comprehensive training data is crucial for improving model accuracy in complex tasks such as mathematical reasoning.

Limitations

While our study successfully constructs the **ViMath-InstructCode** dataset using the **SiRC** framework, it is important to acknowledge some limitations in our approach: i) Firstly, the generalization to other languages of **SiRC**, though promising, is still unclear. The **SiRC** framework and **ViMath-InstructCode** dataset have been specifically designed for Vietnamese. Adapting this framework to other languages, particularly those with even fewer resources, necessitates additional efforts in constructing datasets as well as running experiments; ii) Secondly, the proposed **ViMath-InstructCode** is still prone to noises. The construction of this dataset, despite relying on trusted open sites and undergoing several preprocessing steps to ensure its quality, is completed without any human verification. This could result in a small proportion of faulty samples in our training dataset.

Ethical considerations

Regarding concerns related to the sources of the datasets in our research, they are built from publicly accessible sources, guaranteeing no privacy issues or violations. We do not gather any personally identifiable information, and all data is acquired in compliance with legal and ethical guidelines.

References

Janice Ahn, Rishu Verma, Renze Lou, Di Liu, Rui Zhang, and Wenpeng Yin. 2024. Large language models for mathematical reasoning: Progresses and challenges. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2024: Student Research Workshop, St. Julian's, Malta, March 21-22, 2024*, pages 225–237. Association for Computational Linguistics.

Aida Amini, Saadia Gabriel, Shanchuan Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. 2019. Mathqa: Towards interpretable math word problem solving with operation-based formalisms. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 2357–2367. Association for Computational Linguistics.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Pondé de Oliveira Pinto, Jared Kaplan, Harrison Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Joshua Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021. Evaluating large language models trained on code. *CoRR*, abs/2107.03374.

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

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

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

Yao Fu, Hao Peng, Litu Ou, Ashish Sabharwal, and Tushar Khot. 2023. Specializing smaller language models towards multi-step reasoning. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 10421–10430. PMLR.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Minlie Huang, Nan Duan, and Weizhu Chen. 2023a. Tora: A tool-integrated reasoning agent for mathematical problem solving. *CoRR*, abs/2309.17452.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Minlie Huang, Nan Duan, and Weizhu Chen. 2023b. Tora: A tool-integrated reasoning agent for mathematical problem solving. *CoRR*, abs/2309.17452.

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. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*.

Danqing Huang, Shuming Shi, Chin-Yew Lin, Jian Yin, and Wei-Ying Ma. 2016. How well do computers solve math word problems? large-scale dataset construction and evaluation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics.

Zhenwen Liang, Wenhao Yu, Tanmay Rajpurohit, Peter Clark, Xiangliang Zhang, and Ashwin Kalyan. 2023. Let GPT be a math tutor: Teaching math word problem solvers with customized exercise generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 14384–14396. Association for Computational Linguistics.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 158–167. Association for Computational Linguistics.

Hanmeng Liu, Zhiyang Teng, Leyang Cui, Chaoli Zhang, Qiji Zhou, and Yue Zhang. 2023. Logicot: Logical chain-of-thought instruction tuning. In *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, pages 2908–2921. Association for Computational Linguistics.

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. *CoRR*, abs/2308.09583.

OpenAI. 2023. GPT-4 technical report. *CoRR*, abs/2303.08774.

Sina J. Semnani, Violet Z. Yao, Heidi C. Zhang, and Monica S. Lam. 2023. Wikichat: Stopping the hallucination of large language model chatbots by few-shot grounding on wikipedia. In *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, pages 2387–2413. Association for Computational Linguistics.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. *CoRR*, abs/2302.13971.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, Rui Zheng, Xiaoran Fan, Xiao Wang, Limao Xiong, Yuhao Zhou, Weiran Wang, Changhao Jiang, Yicheng Zou, Xiangyang Liu, Zhangyue Yin, Shihan Dou, Rongxiang Weng, Wensen Cheng, Qi Zhang, Wenjuan Qin, Yongyan Zheng, Xipeng Qiu, Xuanjing Huang, and Tao Gui. 2023. The rise and potential of large language model based agents: A survey. *CoRR*, abs/2309.07864.