Enhancing Numerical Reasoning in Vietnamese Financial Question Answering through Program-Centric Policy Optimization

Duc Dinh Chu[*], **Thanh-Bac Nguyen Ba**[*], **Duy Dinh Le**, **Khanh Van Tran**
FPT Smart Cloud, FPT Corporation
{duccd4, bacnbt, duyld10, khanhtv26}@fpt.com

Abstract

This study details our methodology for the VLSP 2025 Numerical Reasoning QA challenge, focusing on building transparent and accurate models for Vietnamese financial question answering that requires computational reasoning. We propose a two-stage alignment framework combining supervised fine-tuning (SFT) with program-centric policy optimization (PCPO), which is implemented through group relative policy optimization (GRPO) to enhance both program and execution accuracy. First, we leverage an advanced large language model (LLM) to generate high-quality structured reasoning pathways from an augmented dataset derived from the competition organizers' resources. The Qwen3-8B model is then fine-tuned on these structured traces and further refined through GRPO using meticulously designed reward functions to optimize logical consistency. Our approach secured first place among 16 participating teams, achieving 77.87% program accuracy with 82.49% execution accuracy on the public test set, and 76.63% program accuracy with 79.88% execution accuracy on the private test set. Key insights reveal the significance of domain-specific structured reasoning traces, the effectiveness of multilingual data augmentation, and the critical role of PCPO in maintaining accurate numerical reasoning abilities. Our source code and prompts can be found here.

1 Introduction

Numerical reasoning QA in the financial domain represents a critical yet challenging task that requires systems not only to compute correct answers but also to provide transparent and interpretable computation programs. Unlike general QA systems, financial applications demand high precision and explainability, as decisions derived from these systems can significantly impact investment strategies, regulatory compliance, and financial literacy initiatives. The VLSP 2025 Numerical Reasoning QA challenge specifically addresses this need by focusing on Vietnamese financial documents, where models must process complex numerical data embedded within tabular structures and textual contexts to generate both verifiable programs and accurate answers.

Our work contributes to this emerging field by developing a novel framework specifically designed for Vietnamese financial numerical reasoning. We address the dual challenge of achieving high program accuracy while maintaining high execution accuracy through knowledge distillation, chain-of-numerical-reasoning prompt design, synthetic structured reasoning pathway generation, and a two-stage training approach that combines SFT with PCPO. This methodology not only advances the state-of-the-art in Vietnamese financial language processing but also provides insights into effectively adapting large language models for domain-specific numerical reasoning tasks where transparency is paramount.

2 Related Works

Numerical reasoning QA has evolved from general benchmarks that require discrete operations to program-oriented approaches that produce verifiable computation traces. Early datasets such as DROP (Dua et al., 2019) and MathQA (Amini et al., 2019) demonstrated that multi-step arithmetic, counting, and referential reasoning demand models that generate and check structured computation steps rather than relying solely on extractive answers.

In the financial domain, datasets like FinQA (Chen et al., 2021) and TAT-QA (Zhu et al., 2021) adapted this paradigm to real-world reports by pairing questions over hybrid text-and-table inputs with executable programs, exposing challenges such as implicit numeric references, mixed units,

[*]These authors contributed equally.

and multi-step aggregation; these works emphasize programmatic explanations for auditability and trust in financial applications. More recently, DocFinQA (Reddy et al., 2024) augmented 7,437 questions from the existing FinQA dataset with full-document context, thereby extending the average context length from fewer than 700 words in FinQA to 123,000 words in DocFinQA.

Earlier approaches to this task primarily relied on BERT-based models (Devlin et al., 2019). For example, Chen et al. (2021) proposed a system that includes a program generator. In their framework, the retrieval results and the question are first encoded using pre-trained language models. During decoding, at each step, the model can generate tokens corresponding to numbers or table row names from the input, special tokens in the domain-specific language (DSL), or step memory tokens. After completing each operation step, the embeddings of the step memory tokens are updated, enabling the model to retain intermediate reasoning states.

Recent methodological advances combine LLM-driven decomposition with symbolic execution: Chain-of-Thought prompting elicits intermediate reasoning steps, Self-Consistency aggregates multiple reasoning paths to improve robustness, and program-aided approaches (e.g., PAL (Gao et al., 2023)) generate executable code that is run by an interpreter to obtain answers. Complementary data-centric strategies – program-level supervision, cross-lingual transfer, and LLM-driven synthetic augmentation – help adapt models to low-resource domains by reducing annotation needs and aligning numeric/unit formats.

3 Methodology

This section details our comprehensive approach with multilingual data augmentation, structured reasoning trace generation, and precise hyperparameter configurations for a two-stage alignment framework that balances performance and cost to address the dual challenges of program accuracy and execution accuracy in Vietnamese financial numerical reasoning.

Formally, this task is defined as follows: given a financial document containing pre-text, post-text, and tabular data alongside a natural language question, the system must generate a computation program – a single executable operation or a sequence of executable operations – to derive the answer. This paradigm fundamentally diverges from conventional QA by mandating an explicit computation program rather than direct answer prediction.

3.1 Multilingual Dataset Integration

Given the relatively limited quantity of training data provided by the organizers, we initially considered synthesizing additional data. However, synthetic data generation encountered significant obstacles due to the complexity of input data structures and the difficulty in verifying output data quality without extensive human annotation. Consequently, we opted against synthetic data generation and instead chose to augment our dataset with English-language financial data.

After the investigation process, we selected the FinQA dataset, which contains 8,281 samples – more than 2.3 times the size of the VLSP 2025 dataset (3,577 samples) – with input-output patterns directly relevant to our computational reasoning task in the financial domain. Since FinQA is a specialized financial dataset, we decided against translating the FinQA dataset into Vietnamese to preserve semantic integrity, avoid introducing erroneous data that could negatively impact model performance, and conserve computational resources and time.

A critical feature of the FinQA dataset is that beyond providing gold labels comprising program and answer, a substantial portion (approximately 40% of samples) includes an additional field called **program_re**, which contains alternative correct programs for the same question. This characteristic holds significant importance for financial problem-solving as program accuracy plays a crucial role in model learning. The program_re field enhances programmatic diversity in the training process, enabling the model to thoroughly grasp the fundamental nature of financial numerical reasoning tasks and avoid rote memorization. For instance, for the question "Tổng lợi nhuận tích lũy của cổ phiếu phổ thông Citi trong giai đoạn năm năm kết thúc vào ngày 31 tháng 12 năm 2017 là bao nhiêu?", both sequences "subtract(193.5, const_100), divide(#0, const_100)" and "divide(subtract(193.5, const_100), const_100)" represent correct programs as they are mathematically and theoretically equivalent within the financial domain and produce the same answer.

To enhance our training data, we combined FinQA's training, validation, and public test sets, yielding 8,281 samples. For samples containing

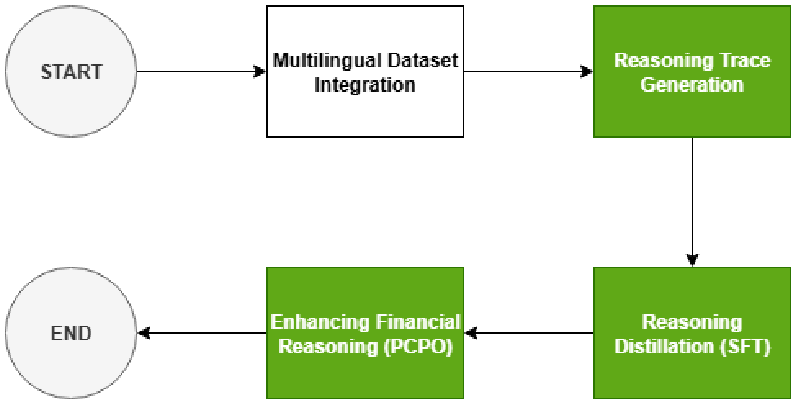
START
Multilingual Dataset Integration
Reasoning Trace Generation
Reasoning Distillation (SFT)
Enhancing Financial Reasoning (PCPO)
END

Figure 1: Overview of the pipeline.

Language	Source	Train	Valid	Public test	Private test
vi	VLSP 2025	2,993	584	497	1,625
en	FinQA	6,251	883	1,147	919
$\sum$		14,661	584	497	1,625

Table 1: Statistics of the VLSP 2025 and FinQA datasets.

the program_re field, we generated an additional sample by replacing the original program with the alternative program, resulting in two distinct samples. This process increased the FinQA-derived portion to 11,688 samples. These were then merged with the 2,993 training samples from VLSP 2025, producing a total of 14,661 training samples. The 584-sample validation set, 497-sample public test set, and 1,625-sample private test set were retained unchanged from the original VLSP 2025 dataset. Details on the number of samples are presented in Table 1.

3.2 Structured Reasoning Trace Generation

3.2.1 Operation Set Formalization

Based on comprehensive analyses of both the VLSP and FinQA datasets, and empirical surveys of real-world computation formulas used by financial experts, we formalize an operation set specifically designed for this challenge. The selection of the operations was guided by 4 key criteria: (1) Covering fundamental arithmetic needed for core financial analysis (e.g., profit calculations, ratio analysis), (2) Supporting essential row-wise aggregations on financial statements (e.g., total annual revenue, average cost) without manual value extraction, (3) Aligning with standard accounting and financial valuation principles, which are primarily based on linear algebra, and (4) Maintaining a balance between generality and feasibility, ensuring the set is composable to handle complex formulas while remaining manageable for program generation.

This formalization is critical for ensuring consistent program generation and enabling rigorous structural evaluation of computational pathways. The mathematical definition of our operation space is presented in Table 2 where $T(r, c)$ denotes the value at row r and column c in the financial table, and C represents the set of relevant columns.

3.2.2 Chain of Numerical Reasoning Prompt Design

To enable the model to generate accurate and reliable programs and answers, it must first possess excellent reasoning capabilities in the financial computation domain to produce *precise structured reasoning traces* before generating the program and answer. We developed a specialized Chain of Numerical Reasoning (CoNR) prompt design that systematically guides language models through structured computational thinking processes. This framework represents a significant advancement over conventional Chain-of-Thought (CoT) approaches by enforcing explicit structured reasoning generation that directly aligns with our primary evaluation metric of program accuracy.

Our methodology draws inspiration from the Program-of-Thought (PoT) approach that recently achieved state-of-the-art results on financial reasoning datasets including FinQA. However, we extended this paradigm with domain-specific adapta-

Operation	Signature	Output	Mathematical Definition
add	n_1, n_2	n	$O_{\text{add}}(n_1, n_2) = n_1 + n_2$
subtract	n_1, n_2	n	$O_{\text{sub}}(n_1, n_2) = n_1 - n_2$
multiply	n_1, n_2	n	$O_{\text{mul}}(n_1, n_2) = n_1 \cdot n_2$
divide	n_1, n_2	n	$O_{\text{div}}(n_1, n_2) = n_1 / n_2$
exp	n_1, n_2	n	$O_{\text{exp}}(n_1, n_2) = n_1^{n_2}$
greater	n_1, n_2	b	$O_{\text{gt}}(n_1, n_2) = (n_1 > n_2)$
table_sum	r	n	$O_{\text{sum}}(r) = \sum_{c \in C} T(r, c)$
table_avg	r	n	$O_{\text{avg}}(r) = \frac{1}{\lvert C \rvert} \sum_{c \in C} T(r, c)$
table_max	r	n	$O_{\text{max}}(r) = \max_{c \in C} T(r, c)$
table_min	r	n	$O_{\text{min}}(r) = \min_{c \in C} T(r, c)$

Table 2: Formalized operation set for financial numerical reasoning.

tions for Vietnamese financial language processing and the stringent requirements of the VLSP 2025 challenge. The full prompt template can be found here.

3.2.3 Structured Reasoning Trace Generation

We used a single node with eight H100 GPUs to serve the advanced LLM and to perform training and inference for all remaining steps. Qwen3-235B-A22B-Thinking-2507 was the open-source model with the highest benchmark score at the time of the competition. We deployed Qwen3-235B-A22B-Thinking-2507 to generate structured reasoning traces for the training set. We identified cases in which the model produced incorrect inferences on the validation set and refined the prompt. This process was repeated until further prompt edits no longer produced a significant increase in accuracy.

The resulting training dataset contained 14,661 samples, and the validation dataset contained 584 samples of structured reasoning traces; these provided a rich training signal for the subsequent SFT stage, effectively transferring the reasoning capabilities of Qwen3-235B-A22B-Thinking-2507 to our target Qwen3-8B model.

3.3 Reasoning Distillation Through SFT

We identified four promising candidate models with up to 13B parameters and trained each model on the structured reasoning traces produced in the preceding stage via a full supervised fine-tuning (SFT) procedure. Importantly, all candidate models were uniformly distilled from the same single teacher model, Qwen3-235B-A22B-Thinking-2507: each candidate was fine-tuned on the identical set of structured reasoning traces generated by that teacher and trained with the same SFT pipeline and hyperparameters (see below). This ensures a controlled and fair comparison of model architectures under the same distillation procedure. This strategy was adopted to distill the high-quality structured reasoning signal into compact architectures while preserving training stability and computational efficiency given our available resources.

The primary training hyperparameters included mixed precision BF16, a maximum sequence length of $5,888$ tokens, a learning rate of 5.0×10^{-5} with a cosine scheduler, 5 training epochs, the AdamW optimizer with 25 warmup steps, and a per-device training batch size of 4. To accelerate training and maximize hardware utilization, we employed DeepSpeed Stage 3 together with FlashAttention 2.

As reported in Table 3, Qwen3-8B and gemma-3-12b-it consistently outperformed the other candidates both before and after SFT; consequently, we selected these two top-performing models for subsequent reinforcement learning using group relative policy optimization (GRPO).

3.4 Program-Centric Policy Optimization via GRPO

We propose Program-Centric Policy Optimization (PCPO) as the high-level, conceptual objective that explicitly prioritizes program structural validity and concision over execution-only rewards. Clarification: PCPO denotes this *objective and reward design*; GRPO is the concrete algorithm we adopt to implement PCPO in our experiments. Below we briefly state the PCPO objective and then describe the GRPO implementation.

Model	Thinking	Before SFT		After SFT	
		PA (%)	**EA (%)**	**PA (%)**	**EA (%)**
mistralai/Mistral-7B-Instruct-v0.3	N	00.00	00.00	49.70	51.71
Qwen/Qwen3-8B	Y	15.09	16.50	**73.84**	**78.07**
deepseek-ai/DeepSeek-R1-Distill-Qwen-7B	Y	3.62	3.86	67.00	71.83
google/gemma-3-12b-it	N	**17.10**	**18.31**	70.02	73.84
Qwen/Qwen3-235B-A22B-Thinking-2507	Y	*30.78*	*33.80*		

Table 3: Evaluation results of candidate models before and after SFT with $temperature$ of 0.6 and top_p of 0.95. PA = Program Accuracy and EA = Execution Accuracy

3.4.1 Reward Function Design

In this competition, program accuracy is the primary evaluation metric, rather than execution accuracy. Therefore, our objective is to design a reward function that balances the two. Specifically, we define the reward as

$$R(p, x) = R_{valid} \cdot (\alpha + \beta \cdot R_{\text{exec}}(p, x) + \gamma \cdot R_{bonus}),$$

with

$$R_{\text{valid}}(p, x) = \begin{cases} 1 & \text{if program is valid,} \\ 0 & \text{if program is invalid } |p| = |p^*|, \end{cases}$$

$$R_{\text{exec}}(p, x) = \mathbb{I}\big[\text{Execute}(p, x) = a^*\big].$$

and

$$R_{\text{bonus}}(p, x) = \begin{cases} 1 & \text{if } |p| < |p^*|, \\ 0.5 & \text{if } |p| = |p^*|, \\ 0.1 & \text{if } |p| > |p^*|, \end{cases}$$

where $|p|$ denotes the number of atomic operations in program p.

We incorporate a validity reward R_{valid} to ensure that all invalid programs are rejected, thereby encouraging the model to generate syntactically correct programs. A valid program receives a fixed score of α. In addition, the **execution reward** $R_{\text{exec}}(p, x)$ is defined as a binary indicator that verifies whether running program p on input x produces the gold-standard numerical answer a^*. Furthermore, we introduce a small bonus term to incentivize concise programs, under the assumption that concision often reflects a clearer and more direct derivation of the underlying financial computation. The overall reward is controlled by three parameters α, β, and γ which balance the contributions of validity, execution, and conciseness, respectively.

The reward computation is implemented as a rule-based verifier and executor that directly re-implements the task's operation set and deterministically evaluates candidate programs against the input data. This design provides a stable, high-fidelity training signal for policy updates: program-level correctness and concision drive the learning objective, while execution-level agreement acts as a necessary but secondary check. By prioritizing program quality in this manner, our optimization process encourages models to produce auditable, logically sound computation traces—an essential requirement for financial numerical reasoning. From the above formulation, increasing α emphasizes outputs that generate correct programs, while increasing β favors outputs that yield correct final results. We selected $(\alpha, \beta, \gamma) = (0.7, 0.2, 0.1)$ after a grid search over $\{0, 0.1, \dots, 1.0\}$ on the VLSP 2025 validation set, using program accuracy as the primary selection metric and confirming stability over multiple random seeds.

3.4.2 Configuration and Outcome

Configuration. The GRPO stage employed vLLM rollouts on a single node with Weights & Biases for experiment tracking. We adopted conservative settings to stabilize policy updates and preserve SFT behaviour, including a learning rate of 1×10^{-6} with the AdamW optimizer and KL regularization with a KL loss coefficient of 0.001. Training used a global batch size of 16, a PPO mini-batch size of 16, and a per-GPU micro-batch size of 2 to manage memory consumption. Rollouts sampled $n = 5$ candidate responses per prompt to obtain stable advantage estimates. Gradient checkpointing was enabled, and the pipeline supported long structured traces (maximum prompt length of 5,888 tokens and maximum response length of 26,880 tokens). Training was conducted for a total of 5 epochs.

Outcome. The experimental results demonstrate that both Qwen3-8B and gemma-3-12b-it achieve substantial performance gains after SFT, with further improvements when incorporating GRPO. Specifically, Qwen3-8B attains 77.87% PA and 82.49% EA under SFT + GRPO, outperforming its SFT counterparts by 4.03 and 4.42 percentage points, respectively. Similarly, gemma-3-12b-it shows consistent enhancements, highlighting the effectiveness of the proposed GRPO framework in optimizing model alignment, as summarized in Table 4.

4 Analysis

4.1 Evaluation Metrics

Program accuracy serves as the primary metric for team ranking, underscoring its importance in financial contexts where auditability of computational pathways is essential. Evaluation is carried out along two dimensions:

- Program Accuracy. This metric assesses whether the generated computation program (e.g., subtract(663, 362)) faithfully reflects the logical structure of the reference solution. It is the sole criterion for competitive ranking, as it ensures both the transparency and correctness of the reasoning process. This requirement is particularly critical in financial applications, where logically flawed programs – even if they yield numerically correct results – pose systemic risks to decision-making support.

- Execution Accuracy. This metric verifies whether the final numerical output (e.g., 26.07) matches the ground-truth value. While informative, execution accuracy does not determine official rankings, as correctness of reasoning takes precedence over coincidental correctness of outputs.

This evaluation framework underscores that in financial reasoning, a correct answer has no meaning or value if the program is incorrect. By prioritizing program accuracy as the definitive benchmark, the challenge enforces strict adherence to auditable, logically sound computational pathways – ensuring models produce solutions that meet regulatory and operational standards for interpretability and reliability in the VLSP 2025 challenge.

4.2 Accuracy Evolution and Comparison

The comparative table of model performance from the untuned baseline through SFT and finally SFT + GRPO – reveals a two-stage improvement pattern. SFT provides the largest baseline uplift by imparting high-quality structured traces that substantially increase both program accuracy and execution accuracy. GRPO then refines program structure: models tend to produce more canonical programs that more closely match reference forms, yielding further gains in program accuracy while preserving or improving execution accuracy. Specifically, Qwen3-8B improved from 73.84% PA / 78.07% EA after SFT to 77.87% PA / 82.49% EA after SFT + GRPO, with comparable trends observed for gemma-3-12b-it. These dynamics indicate that (i) trace quality is the dominant driver at the SFT stage and (ii) relative, verifier-guided reward shaping effectively nudges the model toward auditable reasoning. Operationally, however, GRPO increases computation due to multi-sample rollouts and requires comprehensive verifier coverage to avoid reward-hacking; consequently, stability diagnostics (KL divergence, program-length distributions, and seed variance) should accompany temporal evaluations. Finally, a finer-grained temporal analysis by question type (e.g., aggregation, comparison, ratio) is recommended to pinpoint where SFT versus GRPO contribute most and to guide future targeted improvements.

4.3 Ablation Study

We evaluate the contributions of English data integration and the use of alternative correct programs (via the program_re field) through controlled ablation experiments. As shown in Table 6, training exclusively on Vietnamese data (VLSP 2025) yields a program accuracy (PA) of 66.23% for Qwen3-8B. Incorporating English FinQA samples produces a substantial absolute improvement of 7.61 percentage points in PA; similar trends were observed for gemma-3-12b-it.

These results demonstrate the effectiveness of cross-lingual knowledge transfer for financial numerical reasoning and indicate that exposing models to multiple mathematically equivalent program variants improves their ability to recognize structural invariances in financial computations – a capability that is critical for achieving high program

Model	w/o tuning		SFT		SFT + GRPO	
	PA (%)	EA (%)	PA (%)	EA (%)	PA (%)	EA (%)
Qwen/Qwen3-8B	15.09	16.50	73.84	78.07	**77.87**	**82.49**
google/gemma-3-12b-it	17.10	18.31	70.02	73.84	74.26	78.07

Table 4: Results of Qwen3-8B and gemma-3-12b-it after the GRPO stage on the public test set.

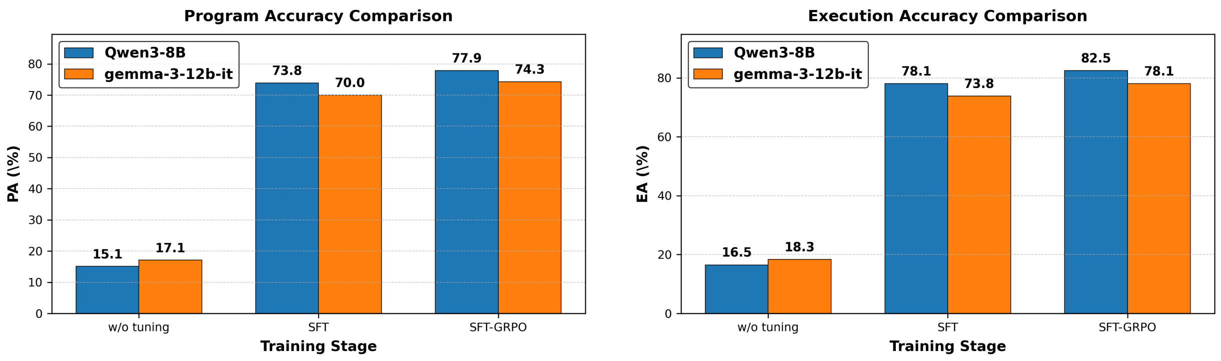

Figure 2: Performance comparison of Qwen3-8B and gemma-3-12b-it across training stages.

Team	PA (%)	EA (%)
HUET	76.63	79.88
ngoquanghuy	75.00	81.95
dathvt	69.82	79.14
truong13012004	69.67	74.26
vietld	61.83	68.49
masterunited	54.14	56.80

Table 5: Ranking on the private test.

Model	SFT			
	VLSP 2025		VLSP 2025 + FinQA	
	PA (%)	EA (%)	PA (%)	EA (%)
Qwen3-8B	66.23	69.25	**73.84**	**78.07**
gemma-3-12b-it	61.97	64.59	70.02	73.84

Table 6: Effect of multilingual data augmentation.

accuracy in real-world applications. Overall, the findings validate our decision to prioritize high-quality multilingual data enrichment with program diversity over synthetic data generation, the latter of which posed substantial verification challenges.

5 Discussion

Our research yields critical insights for developing Vietnamese financial numerical reasoning QA systems. First, the multilingual data integration strategy demonstrated superior effectiveness compared to synthetic data augmentation, where output quality verification encountered insurmountable obstacles. Notably, the program_re field in FinQA proved pivotal in enhancing program structural diversity, enabling models to recognize mathematical equivalence between different expressions - a decisive factor for program accuracy in financial contexts.

Second, the VLSP 2025 challenge results confirm the advantage of optimizing for "program accuracy" rather than solely focusing on numerical outcomes. Models achieving high execution accuracy but low program accuracy (e.g., 81.95% EA with only 75.00% PA) reveal a clear discrepancy between coincidentally correct answers and genuine computational understanding. This distinction is particularly critical in financial applications, where the transparency of decision-making processes holds greater value than final numerical results.

Finally, GRPO proved effective for refining program structures but requires complex reward mechanisms and substantial computational resources. Future work could explore cost-reduction techniques to maintain performance while minimizing resource demands.

6 Conclusion

We propose a two-stage framework to address numerical reasoning challenges in Vietnamese financial question answering, achieving first place among 16 participating teams in the VLSP 2025 challenge with 76.63% program accuracy on the private test set. This study not only establishes a new benchmark for Vietnamese numerical reasoning question answering but also provides a transferable framework for specialized domains requiring transparent decision-making processes.

Future work will focus on researching synthetic data generation for numerical reasoning tasks, expanding the operator set for complex financial calculations, and experimenting with novel scalable reinforcement learning methods to improve accuracy and cost-effectiveness.

Acknowledgments

We would like to thank FPT Smart Cloud for providing the computational resources that enabled the experiments.

Limitations

Our study has several limitations. First, the method relies on GRPO, which is computationally expensive due to multi-sample rollouts and repeated large-model inference, making it less practical in resource-constrained environments. Second, while effective on the VLSP 2025 benchmark, the approach may fail when applied to noisy or out-of-distribution financial documents, such as those with OCR errors or unconventional reporting formats.

References

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, Volume 1 (Long and Short Papers)*, pages 2357–2367, Minneapolis, Minnesota. Association for Computational Linguistics.

Zhiyu Chen, Wenhu Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Huang, Bryan Routledge, and William Yang Wang. 2021. FinQA: A dataset of numerical reasoning over financial data. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3697–3711, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2368–2378, Minneapolis, Minnesota. Association for Computational Linguistics.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Pal: Program-aided language models. *Preprint*, arXiv:2211.10435.

Varshini Reddy, Rik Koncel-Kedziorski, Viet Dac Lai, Michael Krumdick, Charles Lovering, and Chris Tanner. 2024. DocFinQA: A long-context financial reasoning dataset. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 445–458, Bangkok, Thailand. Association for Computational Linguistics.

Fengbin Zhu, Wenqiang Lei, Youcheng Huang, Chao Wang, Shuo Zhang, Jiancheng Lv, Fuli Feng, and Tat-Seng Chua. 2021. TAT-QA: A question answering benchmark on a hybrid of tabular and textual content in finance. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3277–3287, Online. Association for Computational Linguistics.