MoFin: A Small Vietnamese Language Model for Financial Reasoning via Reinforcement Learning

Khac-Duy Nguyen[1,2,3], **Vo Quang Tri**[3], **Le Ho Bao Nhat**[1,2,3],
Dinh Cong Huy Hoang[1,2,3], **Ngo Quang Huy**[3,4], **Huy Tien Nguyen**[1,2,3]

[1] Faculty of Information Technology, University of Science, Ho Chi Minh, Vietnam
[2] Vietnam National University, Ho Chi Minh city, Vietnam
[3] MoMo AI Research Team
[4] SpaceAI
{duy.nguyen28,nhat.le6,hoang.dinh2,huy.ngo,huy.nguyen40}@mservice.com.vn

Abstract

We tackle the VLSP 2025 Numerical Reasoning challenge, which requires answering financial questions in Vietnamese with correct numerical results and transparent reasoning programs. We explore state-of-the-art open language models, including Alibaba's Qwen series and the open Fin-R1 financial model, applying Supervised Fine-Tuning (SFT) and Groupwise Reinforcement Learning from Preferences (GRPO) to teach these models to perform multi-step numerical reasoning. Our approach further augments the training data by translating the English FinQA dataset into Vietnamese and employing a reasoning-optimized model (DeepSeek-R1-Distill-Qwen-7B) to generate high-quality, step-by-step reasoning traces, which are subsequently filtered for correctness. Experiments show that progressively incorporating domain-specific data, majority-voting decoding (Wang et al., 2025) and reinforcement learning significantly boosts both execution accuracy and program correctness. We report notable improvements over baseline models, with the final model MoFin achieving substantially higher accuracy.

1 Introduction

Artificial Intelligence (AI) has advanced rapidly over the past few decades, reshaping a wide range of human activities—including tasks long considered to require uniquely human judgment such as programming, mathematical reasoning, and logical inference. Among the domains most affected is finance and accounting. Natural language processing (NLP) applications, including large language models (LLMs), offers powerful capabilities for reading, interpreting, and acting on complex financial information. However, effective deployment in this domain demands rigorous attention to domain-specific requirements: models must comprehend very long, heterogeneous documents (e.g., narrative text, tables, images, charts), perform numerically exact computations, and provide transparent, auditable outputs that minimize ambiguity in interpretation, extraction, and calculation.

Understanding and analyzing financial reports is a critical yet challenging task, especially when it involves complex numerical reasoning. The VLSP 2025 shared task focuses on Vietnamese financial question answering where systems must not only compute correct answers but also provide an executable reasoning program explaining how the answer was derived.

Formally, given a financial document $\mathcal{D}$ (comprising textual passages and tables) and a Vietnamese question Q, the objective is to output a program $\pi = (o_1, \ldots, o_T)$ drawn from the operation set

$$\mathcal{O} = \left\{ \begin{array}{l} \texttt{add()}, \texttt{subtract()}, \texttt{multiply()}, \\ \texttt{divide()}, \texttt{table\_max()}, \\ \texttt{table\_min()}, \texttt{table\_sum()}, \\ \texttt{table\_average()} \end{array} \right\},$$

such that executing π on $\mathcal{D}$ yields the numeric answer $A \in R$.

In this work, we present our solution to the VLSP 2025 Numerical Reasoning challenge. We investigate the use of small language models (SLMs) for program-guided QA, specifically Alibaba's open-source Qwen-based models, which have strong multilingual and numerical reasoning capabilities, and the domain-specific Fin-R1 model (Liu et al., 2025). Our approach has two main components: **(1) Data Augmentation via FinQA** – to overcome the scarcity of Vietnamese training examples, we leverage the English FinQA dataset (Chen et al., 2021) by translating it and then using a reasoning-optimized SLM to generate detailed reasoning paths for each question. Only logically correct reasoning paths are added to training dataset. **(2) Fine-tuning and Reinforcement Learning**– we apply supervised fine-tuning to teach the model the desired question-program behavior, followed

by a customized RL stage (using the GRPO algorithm) to further align the model's reasoning with correct outcomes, to construct **MoFin** - the small language model tailored for financial reasoning.

2 Related Works

Financial QA Datasets: Our task is closest to FinQA (Chen et al., 2021), which targets numeric, multi-hop reasoning over real financial reports and provides gold solution programs for program generation/execution. Extensions include ConvFinQA (conversational Q&A) and DocFinQA (full annual report contexts). TAT-QA (Zhu et al., 2021) similarly emphasizes numerical reasoning over text–table evidence. We translate FinQA to Vietnamese and use its annotated programs as supervision, reflecting a broader trend toward coupling NLP with symbolic computation.

Financial Domain LLMs: Domain-specific models address finance-specific language and reasoning. Early FinNLP work (e.g., FinBERT (Araci, 2019)) focused on sentiment/NER, while Fin-R1 targets financial reasoning via curated data (including FinQA) and a two-stage SFT+RL pipeline with explicit CoT/reward modeling, reportedly improving interpretability and accuracy. We treat Fin-R1 (7B) as an expert baseline against general LLMs (e.g., Qwen). Relatedly, FinR1 (Liu et al., 2025) distills GPT-4 reasoning to train 7B/14B models. Following this line, we augment training with LLM-generated rationales and fine-tune for financial QA.

3 Methodology

3.1 Overview

Our solution comprises a training pipeline coupled with a data-generation and verification workflow shown in Figure 1. We first translate a total of 5,742 distinct entries in the original FinQA corpus into Vietnamese using OpenAI's o3 model. Each example is then normalized into a single context by converting its table into Markdown and concatenating them with the corresponding pre-text and post-text. This unified representation is fed to reasoning-oriented LLMs—specifically DeepSeek-R1-Distill-Qwen-7B—to produce (i) a chain-of-thought (CoT) reasoning path and (ii) a reasoning program constrained to the operators permitted by the task. The predicted program is executed to yield a numeric answer. For modeling, we conduct Supervised Fine-Tuning (SFT) and Groupwise Reinforcement Policy Optimization (GRPO) on Qwen-based 7B models (Qwen2.5-7B-Instruct, Qwen3-7B-Instruct) and the FinR1 model, selecting the best-performing configuration on the public evaluation set. At inference time, we adopt majority-vote (self-consistency) decoding: the model samples 10 reasoning programs, and we select the most frequent program as the final prediction before execution. Over successive iterations, we refined preprocessing, supervision, and decoding, which improved both execution and program accuracy.

3.2 Data Construction

VLSP 2025 Dataset: The organizers provided a training set of a total of 2,993 sample instances, derived from Vietnamese financial reports (2020–2025) and a translated version of the FinQA dataset. Each example in the dataset consists of a textual context (several paragraphs of pre-text describing company background or financial commentary), a table of financial figures (e.g. quarterly results, ratios, etc.), and possibly some post-text. All programs are represented in a Lisp-like functional format, with operations (*add*, *subtract*, *multiply*, *divide*, etc.) and the ability to refer to intermediate results (denoted as #i for the result of the i-th operation). This formal representation is uniform and executable, which allows automatic evaluation of results. The training data contains the reasoning paths, the ground-truth programs and their executed numeric answers. The private test set (held-out by organizers with 497 sample instances) is used for final evaluation, measuring execution accuracy and program accuracy. We stress that program accuracy requires the model to capture the same reasoning process as the reference, up to commutative reordering. Thus, the model should ideally learn to produce a logically equivalent program even if not identical token-for-token.

Supervised fine-tuning (SFT) trained solely on the original (heavily preprocessed and filtered) VLSP data only attains 60.16% accuracy on the VLSP organizers' public test set, which is not substantially higher than the FinR1 baseline (Table 1). To mitigate data scarcity and domain mismatch while enriching supervision for numerical reasoning in Vietnamese financial QA, we design a human-in-the-loop data pipeline that produces a high-quality Vietnamese corpus with explicit reasoning. The pipeline comprises three stages—data translation, CoT/program generation, and multi-layer validation—summarized in Figure 1.

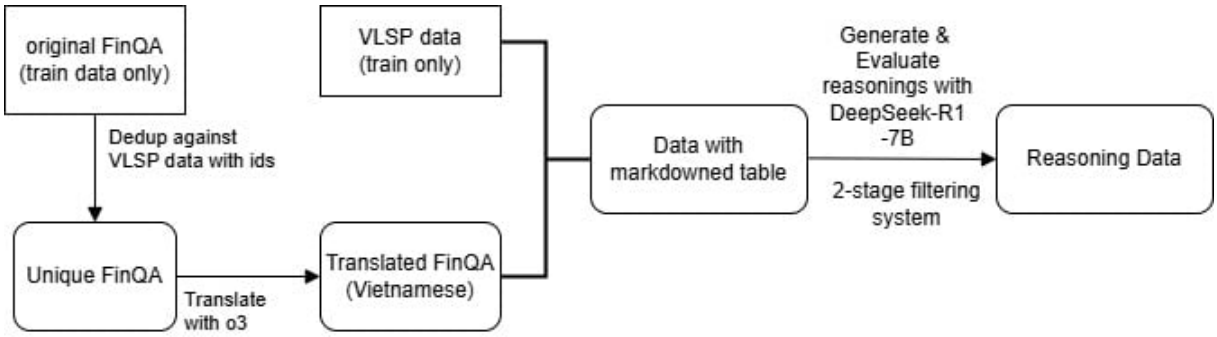

Figure 1: Schematic of the data construction workflow, showing ingestion of raw documents, preprocessing, and assembly into the final standardized dataset.

3.2.1 Data Source

We surveyed FinQA and ConvFinQA, two widely used English-language benchmarks for corporate financial reasoning. During analysis, we observed that many ConvFinQA queries are decompositions of FinQA questions, leading to substantial content overlap between the two sets. Using both would risk data leakage and bias during SFT. Consequently, we restricted external supervision to FinQA (5,742 sample instances) for translation and later merged it with the organizer-provided VLSP corpus to form the comprehensive pool for reasoning-trajectory generation and validation.

For transparency and to facilitate future research, we retain and release the subset of examples that failed our quality filters. Specifically, we identified 670 instances missing a reasoning trajectory and 3,869 instances with malformed program syntax; we also flagged additional cases where the executed numeric answer did not match the FinQA ground truth. This "negative" split has been submitted to the VLSP organizers together with the curated training set, enabling the community to audit, correct, and enrich Vietnamese financial QA resources over time. All items are clearly labeled with failure type and provenance to support systematic curation and error analysis.

3.2.2 Data Processing

(1) Data Translation via FinQA

Because our target domain is Vietnamese finance and accounting, we first evaluated several state-of-the-art English to Vietnamese neural machine translation (NMT) systems—vinai-translate-en2vi-v2, EnViT5-translation, MTet, and PhoMT. In our experiments, these models struggled with (i) domain terminology and (ii) long, mixed-structure contexts (pre-text and tables and post-text), often bounded by relatively short maximum context lengths (e.g., 1000 tokens) and prone to altering Markdown table structure when translating the tables in isolation.

To preserve semantics and structure, we therefore translated the entire 5,742 of FinQA sample holistically—pre-text, table (converted to Markdown), and post-text concatenated into a single context—using OpenAI's o3 model, followed by manual terminology checks aided by bilingual financial dictionaries. Although the gold FinQA programs are largely language-agnostic, we elected to reconstruct the reasoning trajectory and program in Vietnamese rather than directly translating operator strings, ensuring stylistic and structural consistency with VLSP expectations.

(2) Reasoning Trajectory Augmentation

We generated natural-language chains of thought (CoT) and constrained programs in Vietnamese to mirror the style expected in VLSP. Concretely, we used DeepSeek-R1-Distill-Qwen-7B (a reasoning-focused model) as the primary generator, with Qwen2.5-Math-7B-Instruct as a secondary reference when needed. Generation was performed at scale with vLLM, using:

- Temperature was set to 0.6, min p was set to 0.1, maximum new tokens was set to 16,384.
- CoT data wrapped in <think> ... </think> (per Qwen-family recommendations).
- Programs wrapped in <PROGRAM SYNTAX> ... </PROGRAM SYNTAX>, constrained to the task-allowed operators.

Prompts were in Vietnamese and instructed the model to (i) reason step-by-step, (ii) emit a syntactically valid program, and (iii) conclude with an explicit numeric answer. We then parsed the model outputs to derive a pseudo-gold Vietnamese program for each item, normalizing whitespace

and identifiers and aligning the reported numeric answer with the program's result.

(3) Data Validation

Each predicted program was executed and its numeric result compared against the FinQA gold answer. Items failing any criterion—execution mismatch, syntax errors, missing steps, or logically incoherent reasoning—were discarded or regenerated with prompt adjustments. Validation employed a dual-check protocol: Human validation by 5 financial analysts from M Service, focusing on terminology fidelity, table-to-text alignment, and numerical soundness. LLM-as-judge (DeepSeek-R1-Distill-Qwen-7B) to flag structural and logical issues at scale and to standardize format compliance.

This process yielded an augmented set of 3,787 high-quality Vietnamese QA–program pairs derived from FinQA. By filtering strictly for program validity and execution correctness, we ensured the additional data teaches valid solution patterns rather than amplifying noise—akin to prior CoT distillation approaches, but using an open generator better aligned with our domain and deployment constraints.

3.3 Training Method

3.3.1 Training Data Template

Our final training dataset comprised a total of 3,747 distinct entries: (a) 540 samples from the official VLSP Vietnamese financial-report QA with validated Vietnamese CoT and programs , and (b) the translated FinQA subset with validated Vietnamese CoT and programs (3,247 pairs).

We did not directly incorporate additional external datasets; however, FinQA's coverage of common financial-reasoning tasks (e.g., growth-rate calculations, year-over-year comparisons, and segment summations) provided a broad foundation for training. We also reserved the official portion of the original FinQA dataset as a validation set (497 samples) — preprocessed identically to the training data—and combined it with the VLSP organizers' public test set to monitor performance during fine-tuning. Each training instance is rendered as a chat-style serialized sequence with three roles: (1) a **system message** that specifies formatting constraints and response policy (the schema that the model must follow), (2) a **user message** that contains the full context (pre-text, table converted to Markdown, post-text) and the question, and (3) an **assistant message** that provides the reasoning path and the reasoning program. Unless otherwise noted, tokenization follows the base model's native tokenizer, ensuring compatibility with pretrained checkpoints and avoiding undesirable token fragmentation in financial terminology or numeric tokens.

3.3.2 Supervised Fine-Tuning

Given the limited size of the training corpus, parameter-efficient fine-tuning (LoRA) proved decisively more effective than full-parameter updates in our setting. Empirically, LoRA delivered higher validation stability and accuracy while using substantially less compute and memory, likely due to its implicit regularization and better preservation of pretrained knowledge—both of which mitigate overfitting in the low-data regime.

Configurations: We fine-tuned all baseline models with Unsloth's FastLanguageModel, configuring a maximum sequence length of 16,384 tokens with automatic RoPE scaling and training in full-precision BF16 on H100 hardware. Parameter-efficient adaptation used LoRA with rank 128 applied to all projection modules of all layers; lora alpha was set to 256, dropout was omitted, and Unsloth's gradient checkpointing was enabled for long-context training. Tokenization followed the official Qwen-2.5 chat template, which formats each conversation JSON into a single `text` field for supervised fine-tuning. Optimization employed AdamW in 8-bit with a learning rate of 2×10^{-5}, weight decay of 0.01, a linear schedule with five warmup steps, per-step logging. Unless noted otherwise, training used a per-GPU batch of 4 sequences and 4 gradient-accumulation steps, yielding an effective batch of approximately 16 sequences. Models were trained for 3 to 9 epochs over roughly 40000 examples using TRL's SFTTrainer with DataCollatorForSeq2Seq, and all runs used the fixed random seed 3407. Experiments were conducted on two NVIDIA H100 80-GB GPUs in parallel, and a complete run required about three hours for both Qwen2.5-7B-Instruct, Qwen3-7B-Instruct, and FinR1. We monitored the execution accuracy on the validation set after each epoch; both Qwen-based models and Fin-R1 converged to high training accuracy quickly (within one epoch) given the small data size, but we continued into a second epoch with a lower learning rate to refine the program generation format (to reduce syntax errors).

3.3.3 GRPO

After SFT, the model reliably emits well-formed programs; however, some outputs still fail to execute to the correct answer or follow suboptimal reasoning paths (e.g., aggregating three terms in a single step rather than computing intermediate subtotals), which can hinder program equivalence and robustness. Inspired by (Liu et al., 2025), we therefore introduced a reinforcement-learning alignment stage to further improve reasoning fidelity. Specifically, we adopt Groupwise Reinforcement Policy Optimization (GRPO)—a policy-optimization variant that contrasts groups of sampled outputs—well-suited to fine-tuning compact models, and optimize rewards tied to program validity and execution correctness. We also experimented with a length penalty to prefer shorter programs (maximum 7 operations to avoid overly convoluted reasoning), but found the models naturally kept programs concise.

Configurations: The RL training was done using a maximum context of 16,384 tokens, and full-precision weights; for generation within GRPO we use vLLM sampling with minimum probability min-p set to 0.1, top-p set to 1, no top-k cap, a fixed seed of 3407, and stopping at the tokenizer's end-of-sequence, and we produce 4 completions per prompt to enable group-wise relative preference learning. Before training, we tokenize all prompts, compute the empirical 19% of sequence lengths, filter examples exceeding this threshold, set the prompt budget to that percentile plus one token, and allocate the remaining context to completions. We implement 4 reward functions that are applied per group of completions, namely an exact-format reward that grants a higher score when all tags appear correctly and in order, an approximate-format reward that softly encourages single occurrences of each tag while penalizing extraneous ones, an answer-correctness reward that compares the extracted program against the reference answer with both strict and whitespace-insensitive matching, and an answer-correctness reward evaluates the final execution result of the extracted program, granting full credit for an exact match and a positive but smaller reward when the relative error with respect to the ground truth does not exceed 10%. We have bundled all implementation details of the GRPO stage in the source and sent to the VLSP organizers. GRPO hyperparameters are fixed to temperature 2.0, learning rate 5×10^{-6}, weight decay 0.01, warmup ratio 0.1 with a linear schedule, optimizer adamw 8bit, per-device batch size 4 with 4 gradient accumulation, 4 generations per prompt, a cap of 100 training steps with checkpointing at step 100. We performed around 3,500 update steps of RL (which amounted to roughly one pass through the RL dataset) with a KL divergence constraint to ensure the policy doesn't drift too far from the SFT model.

3.4 Evaluation

Evaluation and scoring of model checkpoints are performed automatically on the VLSP organizers' system. The ground-truth annotations for the private test set, including both the reference program syntax and the final execution results, as well as the metric implementations, are kept private by the organizers during and after the competition and cannot be accessed by participants. To ensure objectivity and fairness, all results reported in this paper are taken directly from the final private test leaderboard on the official VLSP website. The public leaderboard reports two metrics as Program Accuracy and Execution Accuracy:

Program Accuracy: For each example in the private test set, the system checks whether the generated program is logically and mathematically equivalent to the reference program. Equivalence is determined by comparing program structures after replacing all concrete arguments with symbolic placeholders and then assessing whether the same sequence of operations is applied, allowing for the following symmetries: reordering of commutative operations, regrouping allowed by associativity, and consistent renaming of intermediate results. Semantically equivalent decompositions are counted as equivalent. A prediction is counted as correct if the generated program is equivalent to the reference program. Program Accuracy is reported as the proportion of examples that satisfy this condition, expressed as a percentage.

Execution Accuracy: For each example in the private test set, the generated program is executed to produce a single numerical answer. This answer is compared with the organizers' ground-truth numerical result for that example. A prediction is counted as correct if the produced answer matches the ground-truth answer under the organizers' comparison procedure. Execution Accuracy is reported as the proportion of examples that satisfy this condition, expressed as a percentage.

3.5 Experiment

In the first stage, the baselines are the off-the-shelf, unfine-tuned checkpoints — Qwen2.5-7B-Instruct, Qwen3-7B-Instruct, and FinR1 — evaluated in a zero-shot setting under the same system prompt and input serialization as all subsequent experiments, ensuring a fair comparison. We experimented with 3 base model configurations:

Fin-R1: a 7-billion-parameter SLM specifically fine-tuned for financial reasoning. Fin-R1 already understands tasks like FinQA and uses a chain-of-thought format with <think> tags for reasoning steps. We used the open checkpoint released by SUFE-AIFLM-Lab (available on HuggingFace). Despite its prior training, Fin-R1 required additional fine-tuning on Vietnamese data, since its core training corpus was primarily English (with some Chinese). We expected Fin-R1 to excel at financial calculations but possibly struggle with Vietnamese language nuances out-of-the-box.

Qwen2.5-7B-Instruct: a 7-billion-parameter general SLM with strong multilingual capability. We chose the instruct variant which is aligned to follow prompts. Qwen-2.5 has demonstrated competitive reasoning ability in the general domain, and we included it to test if a high-capacity general model could adapt to this domain via fine-tuning.

Qwen3-7B-Instruct: the newer generation Qwen3 model at 7B parameters. According to the model card, Qwen3's architecture and training improvements allow a smaller model to match or outperform a larger Qwen2.5 model. Indeed, we observed in our experiments that the Qwen3-7B-Instruct achieved similar accuracy to Qwen2.5-7B on our validation set. We also report results for comparison in Table 1.

4 Analysis

The performance results across successive training stages reveal a clear upward trend for all models, highlighting the effectiveness of both fine-tuning and improved decoding strategies. Our best model, **MoFin**, attains an execution accuracy of **81.95%**, the highest recorded in the VLSP 2025 Numerical Reasoning QA challenge (Table 3). In particular, supervised fine-tuning (SFT) using LoRA adapters produces a substantial jump in accuracy for every model. For instance, Qwen2.5-7B-Instruct's score rises from 0.6016 at baseline to 0.7145 after SFT, an absolute gain of 0.11 (19% relative improvement). Qwen3-7B-Instruct and FinR1 show similar boosts (from 0.60 to 0.71), underscoring that targeted instruction tuning dramatically improves the models' ability to generate correct programs. This large gain is consistent with the known efficacy of LoRA-based fine-tuning, where injecting trainable low-rank matrices into a GPT's layers allows efficient adaptation without full model retraining. The result is a model more aligned to the task distribution, thereby significantly enhancing program correctness out-of-the-box.

Performance Gains across Training Stages: The stepwise improvements from Baseline to SFT, then to GRPO, finally to Majority-Voting Decoding, exhibit diminishing but still meaningful returns at each stage. In summary, across all models:

Supervised Fine-Tuning (LoRA) – provides the largest single-stage gain, raising accuracy by approximately 0.12 absolute (an 18–23% relative increase from baseline).

GRPO Fine-Tuning – yields a further 0.02–0.03 absolute improvement (3–5% relative), indicating moderate gains through outcome-oriented reinforcement learning. GRPO's impact, while smaller than SFT, is notable given that it directly optimizes for correct outputs via verifiable reward signals (e.g. code execution success or math solution correctness).

Majority-Voting Decoding (n=10) – contributes an additional 0.06–0.08 absolute boost (8–11% relative), demonstrating that an ensemble-of-samples decoding strategy can rival or even exceed the benefit of an extra RL fine-tuning stage in improving execution accuracy. This aligns with prior findings (Wang et al., 2025) that self-consistency (majority voting) in CoT decoding yields significant accuracy gains by selecting the most consistent answer among multiple outputs.

These trends indicate that each training intervention addresses different performance gaps: SFT accounts for the bulk of learning the task format and basic reasoning, GRPO fine-tunes the models toward producing verifiably correct outcomes (likely by refining the reasoning or code-generation policy), and the majority-vote decoding then reduces stochastic errors at inference time by aggregating multiple reasoning paths.

Model-Wise Comparison: Comparing the models, we observe that Qwen2.5-7B-Instruct holds a slight but consistent performance edge over Qwen3-7B-Instruct at every stage. For example, after SFT both reach roughly the same tier (0.7145 vs 0.7086), and this gap persists through GRPO (0.7411 vs

Model	Baseline	+ SFT (LoRA)	+ SFT + GRPO	+ SFT + GRPO + MVD (n = 10)
Qwen2.5-7B-Instruct	0.6016	0.7145	0.7411	**0.8195**
Qwen3-7B-Instruct	0.5996	0.7086	0.7396	0.8136
FinR1	0.5795	0.7086	0.7278	0.7885

Table 1: Performance progression across training stages. **MoFin** attains an execution accuracy of 81.95%

0.7396) and final decoding (0.8195 vs 0.8136). The differences (on the order of 0.5–0.8 percentage points) are small, suggesting both Qwen variants benefit similarly from the training pipeline; however, Qwen2.5's marginal lead could imply that its architecture or pre-training gave it a slight advantage in these reasoning-centric tasks. By contrast, FinR1 begins with a lower baseline (0.5795) but nearly catches up to the Qwen models after fine-tuning. In fact, FinR1's score after SFT (0.7086) is on par with Qwen3's, indicating that domain-specific pre-training did not hinder its adaptability. FinR1 is a model originally built for financial reasoning using a two-stage SFT+RL paradigm, yet its baseline underperforms the general-purpose Qwens—likely because FinR1's pre-training was narrower. After GRPO, FinR1 reaches 0.7278, slightly below the Qwens (0.74), and with majority voting it peaks at 0.7885. Thus, even though FinR1 was designed for complex reasoning in a specific domain, the general Qwen models maintain a small absolute advantage (0.02–0.03) in final accuracy on this evaluation. This comparative outcome suggests that a broadly-trained 7B model (Qwen) can match or exceed a specialized model's reasoning performance when both are given equivalent fine-tuning and decoding enhancements. Finally, the majority-voting decoding brings an ensemble effect that substantially boosts execution accuracy without additional training, and elevates our best model (MoFin) to over 81% accuracy.

5 Discussion

Through this project, we gained several insights into building transparent reasoning models in the financial domain, particularly for a language like Vietnamese. We discuss what worked, what did not, and several domain-specific observations below:

Understanding tabular data: We experimented with various methods to handle tables within financial documents. These included converting tables to markdown text, adding descriptive labels for rows and columns, and even preserving the original table format when feeding data into the models (both the base models and those fine-tuned via SFT and GRPO). However, none of these approaches proved completely effective. During evaluation, we still encountered errors in tasks that seemed straightforward, such as extracting the value from a single table cell. This demonstrates that the language models did not truly grasp the two-dimensional relational structure inherent in tables. Moreover, converting tables to a markdown-like format dramatically increased the token count during training and inference, which in turn raised the risk of hallucinations. Understanding and reasoning over tabular data remains a challenging problem – one that is especially important in the financial domain – and likely requires dedicated solutions beyond the techniques we tried.

Fine-tuning specialized vs. general models: We hypothesized that further fine-tuning the FinR1 model (which had already been extensively trained via supervised fine-tuning and Group Relative Policy Optimization, GRPO, on large English and Chinese financial datasets) would allow us to leverage its prior financial knowledge and multilingual capabilities. In theory, this approach would help the model produce well-formatted, accurate reasoning in Vietnamese by exploiting knowledge it learned in other languages. However, our experiments showed the opposite result: performing SFT and GRPO on a general-purpose model yielded better performance than fine-tuning the already finance-specialized FinR1 model. This suggests that a model heavily tuned to a specific domain and language can become over-specialized, making it difficult to adapt to a different language even within the same domain. In our case, starting from a more generic pretrained model and then teaching it the domain and language simultaneously proved more effective than trying to transplant Vietnamese onto an English/Chinese financial expert model.

Persistent output formatting issues: Throughout the evaluation (on both the public and private test sets), we observed numerous output formatting errors, even after the RL fine-tuning phase. Some of the most common issues were: The generated

Frequency of the most frequent answer	10.0	9.0	5.0	7.0	6.0	8.0	4.0	3.0	2.0
Number of samples in private test	587	171	151	144	135	119	119	115	84

Table 2: Distribution stats of majority-voting decoding on the private test set

Rank	Team	Execution Accuracy	Program Accuracy
1	**MoFin**	**0.8195**	0.7500
2	HUET	0.7988	**0.7663**
3	dathvt	0.7914	0.6982
4	truong	0.7426	0.6967

Table 3: Final Leaderboard of VLSP 2025 Challenge on Numerical Reasoning QA. **MoFin** ranks first with the top execution accuracy on the final private test set.

answers sometimes contained anomalous tokens; Duplicate or malformed tokens; Extraneous multiplication in percentages In summary, while RL fine-tuning corrected many aspects of the model's behavior, it did not fully eliminate these format errors. Further refinement (through data cleaning, architectural adjustments, or additional training passes) is needed to address these output artifacts.

Challenges in reward function design: Due to time constraints, we could not extensively explore complex reward designs during the reinforcement learning phase. We implemented only straightforward criteria in our reward function – for example, checking that the output adhered to the desired syntax (and penalizing any malformed format or unwanted tokens), and verifying that the reasoning trajectory and program were present and structured correctly. Designing a reward signal to capture the correctness of the reasoning program, however, proved very difficult. In contrast to tasks where one can simply reward a correct final answer, our task required rewarding the correctness of an entire program (i.e. a sequence of computational steps leading to the answer). Determining whether a generated program is mathematically equivalent to the ground-truth program is non-trivial: two different programs can yield the same result, and operations might be ordered or expressed differently without being wrong. This ambiguity made automatic rewards for "program accuracy" unreliable – we could not feasibly enumerate all equivalent forms of the solution program, and manual evaluation of each output was impractical. In our initial attempt to tackle this, we employed the LLM itself (our DeepSeek-R1-Distill-Qwen-7B model) as a judge to decide if a generated program was equivalent to the reference solution. While this LLM-as-a-judge approach provided some signal, its reliability at scale is uncertain and it may introduce bias. Overall, the difficulty we faced underlines the need for more robust reward function design (potentially involving formal program verification or more advanced automated judges) to properly guide the model during RL fine-tuning.

Hallucinations with partial table operations: Using a majority-vote decoding strategy for chain-of-thought (akin to the self-consistency method of (Wang et al., 2025)), we found a recurring failure on subset queries: when asked to compute over $n-1$ of n values, the model often defaulted to table_sum() or table_average() over the full column, yielding incorrect answers. The converse also occurred: especially in longer programs (four or more ops), the model expanded calculations manually instead of using operators like table_sum(). VLSP scoring may not distinguish correct manual work from operator-based solutions, but both behaviors are undesirable—the first violates constraints; the second signals poor tool use. These patterns reveal gaps in fine-grained logical control and motivate training or architectural changes to enforce subset awareness and appropriate operator selection.

Majority-voting decoding as a self-consistency metric for the SLM: As shown in Table 2, on the private test set (n = 1,625), our best model never produced 10 distinct responses across 10 stochastic decodes; the minimum response frequency was at least 2. The resulting distribution helps identify particularly difficult financial documents. At the same time, although majority-vote decoding substantially boosts performance, it also raises concerns about the consistency of our trained model—suggesting that part of the improvement reflects inference-time ensembling rather than intrinsic single-pass reliability.

Cross-lingual artifacts in reasoning paths:

We often saw Chinese tokens in the model's intermediate reasoning despite Vietnamese-only instructions. Trained first on English/Chinese, the model sometimes switched languages mid-reasoning: the final program was usually in Vietnamese, but traces—and rarely parts of the explanation—appeared in Chinese. This undermines consistency and can confuse evaluators or users, suggesting multilingual pretraining triggers language mixing under hard vocabulary or logic. Enforcing strict language control via decoding constraints is a key next step for production use.

6 Limitations

Despite strong gains, key limitations remain: brittle tabular reasoning (Markdown inflates context and obscures 2-D structure); weak cross-lingual transfer (FinR1 lags general Qwen backbones on Vietnamese, suggesting over-specialization); fragile outputs even with GRPO (spurious tokens, duplication, ambiguous %), requiring deterministic post-processing; hard-to-define RL rewards for program accuracy (equivalence checks are unreliable; LLM-as-judge adds variance); operator misuse (column-wide functions on subsets, manual arithmetic); and majority-vote decoding that masks single-pass instability, with sporadic Chinese leakage in Vietnamese chains.

7 Conclusion and Future Work

In conclusion, our work demonstrated that a combination of high-quality data and careful fine-tuning techniques can successfully adapt a SLM to perform complex reasoning in a specialized domain. Even without access to extremely large models, we achieved strong performance by leveraging open-source models and systematically tailoring them to the task at hand. Working within the Vietnamese financial context introduced several unique challenges—such as handling tabular data and preserving output format integrity—but also offered opportunities for cross-lingual knowledge transfer. In particular, leveraging the English FinQA dataset enabled the model to acquire relevant financial reasoning patterns. Furthermore, this project underscored the importance of evaluation metrics that extend beyond simple accuracy. By optimizing for the correctness of the reasoning program—rather than solely the final numerical answer—we encouraged the model to generate accurate step-by-step solutions, thereby improving the reliability and transparency of its outputs. This emphasis on program-level correctness proved valuable for diagnosing, mitigating errors in the model's reasoning process.

Looking ahead, there are multiple avenues for future work to further improve transparent financial reasoning models. We will strengthen table reasoning with architectures and preprocessing that capture 2-D structure, and develop RL rewards that assess semantic correctness using automated symbolic or LLM-based evaluators. We will also enforce Vietnamese-only reasoning through prompts, decoding constraints, and multilingual training, and pursue cross-lingual adaptation—e.g., vocabulary/embedding transfer or code-switch fine-tuning—to combine finance expertise with multilingual knowledge. Finally, we will reduce hallucinations and remove residual formatting artifacts to improve robustness.

References

Dogu Araci. 2019. Finbert: Financial sentiment analysis with pre-trained language models. *Preprint*, arXiv:1908.10063.

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.

Zhaowei Liu, Xin Guo, Fangqi Lou, Lingfeng Zeng, Jinyi Niu, Zixuan Wang, Jiajie Xu, Weige Cai, Ziwei Yang, Xueqian Zhao, Chao Li, Sheng Xu, Dezhi Chen, Yun Chen, Zuo Bai, and Liwen Zhang. 2025. Fin-r1: A large language model for financial reasoning through reinforcement learning. *Preprint*, arXiv:2503.16252.

Weiqin Wang, Yile Wang, and Hui Huang. 2025. Ranked voting based self-consistency of large language models. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 14410–14426, Vienna, Austria. 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.