Final Project Milestone Report

Graded

Group

Yunchen Yuan
Roger Wang
Jinzi Luo
View or edit group

Total Points

85.6 / 100 pts

Question 1

Feedback **85.6** / 100 pts

✔ **+ 0 pts** Correct

+ 85.6 pts - Are agents applied on top of SFT, or on the base model? Why? Please clarify in the report -- it is not clear at the moment
- Why do some agents hurt performance? Results seem a bit odd, but if that is actually the case, analyze some failure modes.
- RL is a critical part of the project -- we expect to see more progress on it at this point
- How did you decide on the data generation pipeline? Model choice reasoning?
- More details on agents

No questions assigned to the following page.

SLM-Math: Empowering Small Language Models for Mathematical Reasoning

Columbia COMS4705 Project Milestone
Keywords: *Small Language Models, Reinforcement Learning, Mathematical Reasoning, Multi-Agent Systems, Chain-of-Thought*

Roger Wang
Department of Computer Science
Columbia University
lw3240@columbia.edu

Jinzi Luo
Department of Computer Science
Columbia University
jl7199@columbia.edu

Yunchen Yuan
Department of Computer Science
Columbia University
yy3610@columbia.edu

Abstract

We investigate enhancing small language models for mathematical reasoning through a four-stage pipeline: (1) CoT data generation via Grok-4.1 and MiniMax-M2, (2) supervised fine-tuning (full and LoRA) on Qwen2.5-Math-1.5B, (3) GRPO reinforcement learning, and (4) ten agentic workflow architectures. Results demonstrate substantial gains from both training-based improvements and inference-time multi-agent verification, with cross-dataset transfer from GSM8K to MATH and consistent benefits from Solver-Checker architectures.

1 Key Information

- TA mentor: Melody Ma
- External collaborators: No
- Sharing project: No

2 Approach

2.1 Baseline Model

We employ Qwen2.5-Math-1.5B [1] as our base model, a small language model with 1.5 billion parameters that has been pretrained on extensive mathematical corpora including textbooks, problem sets, and solution traces. This domain-specific pretraining provides strong mathematical foundations while maintaining computational efficiency suitable for our multi-stage training pipeline.

2.2 Implementation Pipeline

We implement a four-stage pipeline combining training-based improvements with inference-time agentic workflows.

2.2.1 Data Pipeline

We generate high-quality CoT data using a two-round cascade: Grok-4.1-Reasoning-Fast followed by MiniMax-M2 with multiple attempts. Each attempt applies configurable thinking effort (300–2500

No questions assigned to the following page.

tokens) with structured output format: `<think>...</think>` reasoning followed by solution steps ending in `\boxed{answer}`. Only solutions matching ground truth answers are retained. From 19,473 training samples (7,473 GSM8K [2] + 12,000 MATH [3]), we obtain 18,946 verified CoT samples, filtering 527 unsolved problems (2.71%).

2.2.2 Supervised Fine-Tuning

We train Qwen2.5-Math-1.5B using GSM8K CoT data in two configurations: (1) **Full SFT**: LR 5×10^{-5} (2) **LoRA (r=16)** [4]: LR 1×10^{-4}. Both use cosine scheduling and bfloat16, trained on batch size 128 with 2 epochs . Training exhibits rapid convergence and demonstrates transfer from GSM8K to MATH-500, suggesting general reasoning improvement rather than memorization.

2.2.3 Reinforcement Learning

We apply GRPO [5] to the SFT LoRA checkpoint with binary rewards ($r = \pm 1.0$) and KL regularization ($\beta = 0.05$): $\mathcal{L}_{\text{GRPO}} = -\mathbb{E}[r \cdot \log \pi_\theta] + \beta \, \text{KL}(\pi_\theta \| \pi_{\text{ref}})$. Training uses $K = 2$ responses/prompt, batch 16, LR 5×10^{-6}, 1 epochs on GSM8K. Policy optimization provides further gains beyond SFT by directly rewarding correct answers.

2.2.4 Agentic Workflows

We implement nine training-free inference-time agent architectures [6, 7]: (1) **Solver-Checker** variants (Stateless, Chat, Summarizer, With-Tools) use iterative verification loops where a solver generates solutions and a checker validates them, with variants differing in conversation management (stateless uses independent prompts; chat-based maintains dialogue history), optional summarization layers to reduce context noise, and tool execution for code-based numerical verification. (2) **Majority Vote** aggregates multiple inference runs—first deterministic, then with different random seeds—via voting with first-run preference. (3) **Python Tools** is a single-shot agent that executes generated code blocks for precise computation without iterative verification. All agents share deterministic first-round decoding for fair accuracy comparison.

3 Experiments

Datasets: We use 500 samples from GSM8K test set and Math-500 dataset for evaluation. **Metrics:** Pass@1 accuracy.

Results

Configuration	GSM8K-test	MATH-500
Base Model		
Qwen2.5-Math-1.5B	65.8%	53.2%
Training-based Methods		
SFT (LoRA r=16)	80.0%	**67.2%**
SFT (Full)	**81.6%**	67.0%
Agentic Workflows		
Solver-Checker With-Tools	81.4%	49.8%
Majority Vote	70.2%	54.8%
Agent with Python Tools	72.6%	45.2%

Table 1: Pass@1 accuracy across training methods and agent architectures. GSM8K-test uses 500 samples.

Analysis

Our results demonstrate substantial improvements through both training and inference-time interventions. SFT yields rapid convergence with both LoRA and full fine-tuning producing comparable results. Cross-dataset transfer from GSM8K to MATH-500 indicates general reasoning improvement.

No questions assigned to the following page.

GRPO training is ongoing due to its higher computational demands—generating multiple candidate responses per prompt and computing reward signals. Agentic workflows, particularly Solver-Checker with Tools, Majority Vote, and Agent with Python Tools outperform the baseline by leveraging external computation delegation and ensemble robustness. Other tested workflows (summarizer variants, stateless checkers, summarizer) exhibited performance degradation (gsm8k-test 46.8% to 59.4%.) due to: insufficient model parameters for complex multi-agent coordination, limited context windows constraining elaborate prompts, and over-specialized role designs reducing cross-task generalization.

4 Future Work

Our future directions include: (1) **Training Optimization and Data Expansion**: optimizing training configurations and hyperparameters, expanding to full MATH dataset and competition problems (AIME, AMC) [8]; (2) **Advanced RL Methods**: improving reward mechanisms through fine-grained rule design (formatting, step validity), reward models, and process supervision; exploring advanced sampling strategies integrating Monte Carlo Tree Search (MCTS) [7] for more effective exploration; (3) **Agentic RL** (most important): training models through agentic workflows combined with domain expert models [9], enabling joint optimization of multi-agent reasoning and policy learning to achieve synergistic improvements beyond independent training and inference-time augmentation.

5 Response to TA Feedback

Q1: How do we blend SFT and multi-agent RL? We first apply SFT to establish stable reasoning behavior using high-quality CoT traces, providing a warm initialization. Subsequently, we introduce GRPO reinforcement learning to optimize the policy through iterative sampling and outcome-based rewards, building upon the SFT foundation while enabling exploration of diverse solution strategies.

Q2: Where do teacher traces come from? Our teacher traces are generated by frontier reasoning models (Grok-4.1-Reasoning-Fast and MiniMax-M2) through our two-round data pipeline. These external expert models produce step-by-step solutions with explicit reasoning processes, which are verified against ground truth answers before being used as training data for the student model.

References

[1] An Yang, Beichen Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*, 2024.

[2] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[3] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

[4] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

[5] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y Li, Y Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

[6] Yiran Wu, Feiran Jia, Shaokun Zhang, Hangyu Li, Erkang Zhu, Yue Wang, Yin Tat Lee, Richard Peng, Qingyun Wu, and Chi Wang. Mathchat: Converse to tackle challenging math problems with llm agents. *arXiv preprint arXiv:2306.01337*, 2023.

[7] Xinyu Guan, L. Zhang, Yifei Liu, Ning Shang, Youran Sun, Yi Zhu, Fan Yang, and Mao Yang. rstar-math: Small llms can master math reasoning with self-evolved deep thinking. *arXiv.org*, 2025.

No questions assigned to the following page.

[8] Maxwell Jia. AIME_2024: A dataset of 2024 American Invitational Mathematics Examination problems with solutions and answers. https://huggingface.co/datasets/Maxwell-Jia/AIME_2024, 2024. Dataset at Hugging Face, tag: explanation-generation, license: MIT.

[9] Chanwoo Park, Seungju Han, Xingzhi Guo, A. Ozdaglar, Kaiqing Zhang, and Joo-Kyung Kim. Maporl: Multi-agent post-co-training for collaborative large language models with reinforcement learning. In *Annual Meeting of the Association for Computational Linguistics*, 2025.