SMOLSolver: Lightweight Generator–Verifier Framework for Step-by-Step Math Problem Solving

Akshat Singh **Zhuwei Xu** **Shiyue Zhang** **Raj Trikha** **Nishant Sharma**
New York University
{as20255, zx2188, sz5331, rt2932, ns6287}@nyu.edu

Abstract

This intermediate report presents ongoing work on **SMOLSolver**, a lightweight two-model framework for step-by-step arithmetic problem solving. The system combines a **Generator**, which produces detailed, multi-step solution traces for mathematical problems and a **Verifier**, which evaluates each reasoning step to enable future iterative refinement. Both models are small language models fine-tuned on standard math-reasoning datasets. The project aims to study how verifier-guided feedback can enhance reasoning consistency and numeric accuracy without relying on large model sizes. We summarize fine-tuning progress for both components, initial results, and the next steps toward integrated inference with the feedback loops.

1 Introduction

SMOLSolver aims to develop an efficient, modular math-solving system composed of two independently tuned components:

- **Generator:** Produces detailed, multi-step reasoning traces for arithmetic problems, breaking down the solution into interpretable steps.
- **Verifier:** Assesses the logical and numerical correctness of each step and provides targeted textual feedback for iterative refinement.

The system targets the GSM8K dataset and similar reasoning benchmarks while remaining lightweight enough for single-GPU fine-tuning.

1.1 Motivation

Large models achieve strong reasoning ability but require high compute and memory [Lightman et al., 2023]. This project investigates whether smaller models ($\leq$ 3B parameters) can approximate such performance through dataset-specific LoRA fine-tuning and modular design.

2 Literature Review

Recent advancements in mathematical reasoning have increasingly focused on process supervision over outcome-only evaluation. While large models demonstrate high performance, a parallel line of research has confirmed the capabilities of small language models (SLMs). Projects like Instella-Math and Arithmo2 show that SLMs can achieve strong results on benchmarks like GSM8K, especially when combined with parameter-efficient fine-tuning (PEFT) methods. Techniques such as QLoRA, in particular, have shown near full-finetuning performance on these tasks [Dettmers et al., 2023], validating our project's foundational use of lightweight models.

Our **SMOLSolver** framework builds on this foundation by exploring a modular, verifier-guided architecture as a compute-efficient alternative to large-scale, monolithic training. This approach is contextualized by recent work on feedback-driven refinement. For instance, StepCo [Wu et al., 2025] introduces stepwise correction signals to iteratively refine answers, and SIPF [Chen et al., 2025] proposes structured feedback injection specifically for small models. Both studies demonstrate measurable accuracy gains using just a single feedback loop, which directly motivates our project's core design: a "one-shot" correction pipeline where a dedicated verifier model guides the generator to improve reasoning.

3 Methodology

We propose **SMOLSolver**, a lightweight dual-model framework designed to enhance mathematical reasoning in small language models through *collaborative verification*. The system consists of a **Generator** (*Phi-2*, 2.7B) that produces step-by-step reasoning chains, and a **Verifier** (*TinyLLaMA*, 1.1B) that evaluates the correctness of each rea-

soning step. We fine-tune both models independently using parameter-efficient methods: the Generator on *GSM8K* solution data and the Verifier on *PRM800K* step-level labels. The integration of these components into a *self-correcting feedback loop* is planned as future work.

3.1 System Overview

SMOLSolver uses a three-stage pipeline: *Generation*, *Verification*, and *Refinement*. Given a problem q, the *Generator* produces a step-by-step solution $r = G(q)$. The *Verifier* then outputs step-level correctness labels $v = V(q, r)$, which will be fed back to the *Generator* to produce a corrected solution. The *Generator* is trained on **GSM8K** and the *Verifier* on **PRM800K**, allowing for modular training and sequential inference.

3.2 Generator Module

The *Generator* is a **Phi-2** (2.7B) model fine-tuned on the **GSM8K** dataset using **QLoRA** (4-bit quantization). It is trained to produce numbered reasoning steps ending with "`FINAL_ANSWER: <number>`" for parsability. We use instruction fine-tuning with prompt masking (loss on response tokens only) and apply **LoRA** (rank=16) to attention layers for parameter-efficient training.

3.3 Verifier Module

The *Verifier*, a **TinyLLaMA** (1.1B) model, classifies reasoning steps as *wrong*, *unclear*, or *correct*. It is fine-tuned using **LoRA** on the **PRM800K** dataset, which provides step-level correctness scores $\{-1, 0, 1\}$. We use a two-phase pipeline: **Phase 1** trains on simple, short-context examples to learn basic label mapping. **Phase 2** continues training on longer, more complex reasoning chains to refine its ability to evaluate multi-step deductions. LoRA is applied throughout both phases for efficiency.

4 Experiments and Results

4.1 Generator Fine-Tuning

4.1.1 Setup

- **Model:** `microsoft/phi-2` (2.7B parameters)
- **Method:** QLoRA (4-bit quantization, fp16)
- **Objective:** Generate structured step-by-step mathematical reasoning with explicit `FINAL_ANSWER:` delimiter for parseable numeric outputs.
- **LoRA Config:** rank=16, α=16, dropout=0.1
- **Target Modules:** q_proj, k_proj, v_proj, o_proj Cosine learning rate schedule with 3% warmup for stable convergence across 2 epochs.
- **Hyperparameters:** Epochs=2, Batch Size=32 (effective), LR=1×10^{-4}, Max Length=1024 tokens, Precision=FP16

4.1.2 Results

Pass@1 Accuracy: Proportion of problems solved correctly with single deterministic solution (temperature=0).

Format Compliance: Percentage of outputs with valid parseable `FINAL_ANSWER:` line.

Metric	Result
Pass@1 Accuracy	64.06%
Correct / Total	476 / 743
Format Compliance	99.33%
Validation Loss (final)	0.486
Trainable Params	7.86M (0.28%)

Table 1: Generator performance on GSM8K validation (748 examples).

The generator achieved 64.06% Pass@1 accuracy, substantially outperforming zero-shot Phi-2 and competitive with GPT-3.5 ($\sim$57%). Majority@5 sampling improved accuracy by +3.05%, indicating the model frequently produces correct answers but benefits from reasoning diversity. Near-perfect format compliance (99.33%) confirms reliable template learning. Ablation studies showed increasing LoRA rank to $r = 32$ yielded no improvement (64.46%), indicating performance is limited by reasoning quality rather than capacity.

4.1.3 Example

Question: Mimi picked up 2 dozen seashells. Kyle found twice as many. Leigh grabbed one-third of Kyle's shells. How many did Leigh have?

Generated:

```
1) Mimi has 2 × 12 = 24 shells.
2) Kyle has 2 × 24 = 48 shells.
3) Leigh has 48 / 3 = 16 shells.
FINAL_ANSWER: 16
```

Ground Truth: 16

4.2 Verifier Fine-Tuning

4.2.1 Setup

- **Model:** `TinyLlama/TinyLlama-1.1B-Chat-v1.0`

- **Method:** LoRA fine-tuning (8-bit, fp16)
- **Objective:** Multi-class classification of reasoning step validity (`wrong/unclear/correct`) based on generated step-by-step solutions.
- **Hyperparameters:** Epochs=2, Batch Size=1, LR=1×10^{-5}, Max Length=512 tokens

4.2.2 Results

Accuracy: The proportion of reasoning steps whose predicted validity label (`wrong/unclear/correct`) matches the human-annotated ground truth.

Macro F1: To balance label imbalance and evaluate overall step consistency.

Evaluation Loss: Cross-entropy loss on the held-out test set.

Phase	Accuracy	F1 (macro)	Eval Loss
Phase 1	0.8791	0.3119	0.4293
Phase 2	0.8249	0.3060	0.7253

Table 2: TinyLlama verifier performance across two fine-tuning phases.

The verifier achieved strong verdict alignment, showing that it reliably distinguishes correct, uncertain, and erroneous reasoning patterns. Performance slightly declined in Phase 2 due to increased reasoning length and complexity.

4.2.3 Example

Step Verification

```
=== Step Verification ===
Mimi has 24 shells.                    → correct
Kyle has 48.                           → correct
Leigh has 48/3 = 16.                   → correct
Answer: 16.                            → correct
```

4.2.4 Alternative Model Exploration (Phi-2)

In addition to the `TinyLlama` model, the team also explored `microsoft/phi-2` as a verifier backbone. These experiments used QLoRA and tested various configurations, including LoRA ranks (`r=16` and `r=64`) and data sizes (10% and 20% samples).
This exploration found that the `phi-2` verifier consistently achieved a stable evaluation accuracy of ∼**81.5%** (`eval_mean_token_accuracy`). While increasing LoRA capacity and data size improved training accuracy up to 84.8%, it did not significantly improve the final validation accuracy over the baseline. This performance was notably lower than the `TinyLlama`'s 87.9% accuracy, reinforcing the choice to proceed with `TinyLlama` as the primary verifier model.

5 Current Progress

The project is a team effort, with responsibilities divided by component. The Generator module, including all fine-tuning on the GSM8K dataset and subsequent analysis, is managed by Akshat Singh, Zhuwei Xu, and Raj Trikha. The Verifier module, which involves data processing of the PRM800K dataset and fine-tuning experiments, is managed by Shiyue Zhang and Nishant Sharma. This report presents the individual tuning results from both subteams. The final milestone will involve all members collaborating to integrate these two components into a unified feedback loop.
The next and final planned milestone is the **Joint inference evaluation**, where the best-performing Generator and Verifier models will be integrated to assess the accuracy improvements from the feedback loop.

6 Future Plan

The next stage integrates the generator and verifier for automated inference. The verifier's feedback will determine whether generator outputs are accepted or flagged for re-generation. We also plan to compare `Phi-2` and `Phi-3-Mini` verifier backbones for trade-offs between accuracy and compute.

References

Kaiyuan Chen, Jin Wang, and Xuejie Zhang. 2025. Learning to reason via self-iterative process feedback for small language models. In *Proceedings of the 30th International Conference on Computational Linguistics (COLING)*, pages 2369–2383.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms. *arXiv preprint arXiv:2305.14314*.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. *arXiv preprint arXiv:2305.20050*.

Zhenyu Wu, Qingkai Zeng, Zhihan Zhang, Zhaoxuan Tan, Chao Shen, and Meng Jiang. 2025. Enhancing mathematical reasoning in llms by stepwise correction. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 18845–18861.