Midterm Kaggle Competition

Khwaab Thareja[†], Rachit Pathak[†], Xiaozhou Wen[†]
New York University
Deep Learning - Fall 2025
{kt3180, rmp10015, xw3795}@nyu.edu

Abstract

This report presents our approach to the Math Question Answer Verification Competition, where we supervised fine-tune a Llama-3 8B model to predict the correctness of mathematical answers. We detail our methodology including data preprocessing, LoRA-based parameter-efficient fine-tuning, prompt engineering strategies, and extensive hyperparameter optimization. Our experiments demonstrate Our experiments demonstrate a 16.1% relative improvement over the 0.726 baseline, achieving a final test accuracy of **84.288%**. We analyze what techniques proved effective and discuss the challenges encountered. Code and model weights are available at xiaozhoudev/CS-GY-6953-DL-Midterm.

1 Introduction

Automated verification of mathematical answers is essential for intelligent tutoring systems and educational platforms. This competition challenges us to build a binary classifier using Llama-3 8B that determines whether a given answer to a math question is correct or incorrect.

Large Language Models (LLMs) have shown remarkable capabilities in mathematical reasoning [1], making them ideal candidates for this task. However, fine-tuning such large models requires efficient techniques due to computational constraints. We employ Low-Rank Adaptation (LoRA) [2] to make fine-tuning tractable while maintaining performance.

Our Approach: Our main strategy involved iterative experimentation, focusing on data volume, class balancing, and prompt engineering. We found that a combination of undersampling the majority class for a balanced dataset and using a role-based 'teacher' prompt—which explicitly included the 'question', 'solution' and 'answer' fields—yielded the best results.

Contributions: Our key findings are:

- We demonstrate that prompt engineering (specifically, adding a 'teacher' persona and including the separate 'answer' field) provided the most significant accuracy boost, jumping from 81.754% to 84.148%.
- We show that training on a larger, class-balanced dataset (up to $\sim$40,000 samples) consistently improved performance.
- We identify an effective set of hyperparameters (cosine LR scheduler, $r = 128$, and zero LoRA dropout) for this task.

2 Dataset

2.1 Dataset Description

The dataset consists of mathematical questions paired with answers and correctness labels. Each instance contains: (1) *question*: the math problem, (2) *answer*: provided solution (correct/incorrect), (3) *solution*: detailed reasoning, and (4) *is_correct*: binary label (True/False).

2.2 Data Analysis

The full dataset provides 1,000,000 training samples and 10,000 test samples. A preliminary analysis of a 50,000-sample subset revealed an imbalance: 30,033 (60.1%) samples were labeled 'False' and 19,967 (39.9%) were 'True'. To address this, we used 1:1 balanced undersampling. Our final model was trained on a 50,000-sample subset, which was undersampled to a balanced 39,934 samples. We used a 2,000-sample slice for validation. We set a `max_seq_length` of 6144 tokens to accommodate lengthy problem descriptions and solutions.

2.3 Preprocessing

We applied several preprocessing steps. We implemented a `clean_solution_text` function to remove HTML tags (e.g., '<llm-code>') and LaTeX artifacts (e.g., `'\boxed{}'`) from the solution text to normalize the input. We constructed our final prompt by combining the 'question', the cleaned 'solution', and the 'answer' text, as shown in Section 3.2.

3 Methodology

3.1 Model Architecture

Llama-3 8B: We use Llama-3 8B as our base model, a state-of-the-art transformer with 8 billion parameters featuring improved tokenization and extended context length capabilities.

LoRA Fine-Tuning: Given memory constraints, we employ Low-Rank Adaptation (LoRA) which introduces trainable low-rank matrices into attention layers while freezing pre-trained weights. This dramatically reduces trainable parameters from 8B to approximately 336M parameters.

Our LoRA configuration: rank $r = 128$, alpha $\alpha = 256$, dropout $= 0.0$, targeting all 7 key modules: `q_proj`, `k_proj`, `v_proj`, `o_proj`, `gate_proj`, `up_proj`, and `down_proj`.

We apply 4-bit quantization using 'bitsandbytes' to further reduce the memory footprint, enabling us to train on a single GPU.

3.2 Prompt Engineering

We formulated the task as an instruction-following problem. Our final, most successful prompt (used in Exp 4 & 5):

```
You are a teacher with great
mathematician skilled in verifying
mathematical solutions.Your task is
to determine if the student's solution
correctly solves the math question.
Below is the teacher's question and
stuent's solution.

Teacher's Question:
{}.

Student's Solution:
{}.

Student's final answer:
{}.

Output 'True' if the solution is fully
correct (sound reasoning, correct logic,
and accurate final answer), otherwise
'False'.

Output:
{}
```

Our key finding (Exp 3 vs. Exp 4) was that including the 'answer' field separately from the 'solution' text was critical. Our initial prompt (Exp 1-3) only used 'question' and 'solution', yielding a max score of 81.754%. The new 'teacher' prompt, which also formatted the 'answer' field, immediately boosted this to 84.148%.

3.3 Training Configuration

Table 1 shows our final training hyperparameters. We used a cosine scheduler for smooth convergence and found that a `learning_rate` of 6e-5 with a small `weight_decay` of 0.005 worked best. LoRA dropout was set to 0, as it improved performance over 0.1 in our experiments.

Hyperparameter	Value
Learning Rate	6.0e-5
Batch Size	16
Gradient Accumulation	4
Effective Batch Size	64
Epochs	5
Max Sequence Length	6144
Warmup Steps	10
Weight Decay	0.005
Optimizer	AdamW (8-bit)
LR Scheduler	cosine
LoRA Rank (r)	128
LoRA Alpha (α)	256
LoRA Dropout	0.0

Table 1: Final model (Exp 5) hyperparameters

We used cross-entropy loss and trained for 5 epochs on an NVIDIA A100 GPU. The final training run took approximately 8.5 hours.

4 Experiments and Results

4.1 Baseline Performance

We first evaluate the provided baseline model, which achieves 0.726 test accuracy.

4.2 Experimental Setup

We conducted 5 main experiments, iteratively building on our findings.

- **Exp 1:** Baseline (8k data, 5 epochs, 5e-5 LR, BSize 32).
- **Exp 2:** Exp 1 + BSize 16 (doubled update steps).
- **Exp 3:** Exp 2 + More data (∼16k samples, 3 epochs).
- **Exp 4:** Exp 3 + More data (∼24k, 5 epochs), new 'Teacher' prompt, & new hyperparams (6e-5 LR, cosine, 0 dropout).
- **Exp 5 (Final):** Exp 4 + More data (∼40k samples).

4.3 Results

Table 2 summarizes our results. The two major improvements came from the prompt/hyperparameter change in Exp 4 and the data increases. The final model (Exp 5) achieved 85.50% validation accuracy and 84.288% on the private Kaggle leaderboard.

Configuration	Val Acc.	Test Score
Provided Baseline	-	0.72600
Exp 1: 8k data, BSize 32	83.40%	0.79742
Exp 2: 8k data, BSize 16	83.20%	0.80647
Exp 3: 16k data, 3 Epochs	83.25%	0.81754
Exp 4: 24k data, New Prompt	84.85%	0.84148
Exp 5: 40k data, Final	**85.50%**	**0.84288**

Table 2: Experimental results comparison

4.4 What Worked

- **Prompt Engineering:** Changing to our final 'teacher' prompt and *including the 'answer' field* was the most impactful change, responsible for the jump from 81.754% to 84.148%.
- **Data Scaling & Balancing:** Increasing the training data from 8,000 to ∼40,000 samples and applying 1:1 class balancing via undersampling steadily increased performance.
- **Hyperparameter Tuning:** Switching to a 'cosine' LR scheduler and removing `lora_dropout` (setting to 0.0) were critical for the Exp 4 breakthrough.

4.5 What Didn't Work

- **Initial Batch Size:** Our first experiment with a batch size of 32 (Exp 1) performed worse (79.742%) than the identical setup with a batch size of 16 (Exp 2, 80.647%). This suggests that more frequent gradient updates were beneficial.
- **LoRA Dropout:** We found that `lora_dropout = 0.1` (used in Exp 1-3) consistently underperformed `lora_dropout = 0`. Regularization via LoRA dropout was not necessary for this task.

4.6 Error Analysis

From the validation report of our final model (Exp 5), we achieved 85.50% accuracy. The model had higher recall for 'False' (90%) but lower recall for 'True' (78%). This means the model is excellent at identifying incorrect solutions but is more likely to misclassify a *correct* solution as 'False'. These false negatives (correct answers marked as false) appear to be the main source of error, suggesting the model is overly critical or gets confused by correct-but-unconventional reasoning paths.

5 Discussion

Our final model achieves 84.288% accuracy, representing a 16.1% relative improvement over the baseline. Our systematic approach isolated prompt engineering and data scaling as the two most important factors. The 'teacher' prompt likely framed the task more effectively. Providing the 'answer' field separately allowed the model to directly compare the reasoned 'solution' against the provided 'answer', rather than trying to infer the answer *from* the solution text.

Challenges: The main challenge was the long training time, with our final model taking over 8.5 hours. This limited the number of full-scale experiments we could run. Another challenge was the dataset imbalance, which we addressed with undersampling. This was effective but meant we discarded a large portion of the majority class data.

6 Conclusion

We successfully fine-tuned Llama-3 8B for math answer verification using LoRA. Our iterative approach of data scaling, class balancing, and prompt engineering resulted in a final Kaggle score of 84.288%. The most critical factor was a prompt redesign that included the ‘answer‘ field and adopted a ’teacher’ persona.

Future Work: Future work could explore training on the full 1M dataset, experimenting with different class balancing techniques (e.g., oversampling, T5-based pre-training), or ensembling models trained on different data subsets.

References

[1] Meta AI. Llama 3 Model Card. `https://github.com/meta-llama/llama3`, 2024.

[2] 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. In *International Conference on Learning Representations (ICLR)*, 2022.

[3] Unsloth AI. Unsloth Library. `https://github.com/unslothai/unsloth`, 2024.

Acknowledgments

We utilized Google Colab with NVIDIA A100 GPUs for model training, the Unsloth library for memory-efficient 4-bit fine-tuning and used chatGPT to learn the unsloth library and the hyperparameter for fine-tuning.