

Midterm Kaggle Competition

Lyuyongkang Yuan[†], Junwei Yan[†]
New York University
Deep Learning - Fall 2025
{ly2188, jy4831}@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 a moderate improvement (about 10%) over the baseline within a quantization model and small LoRA rank, achieving a final test accuracy of **0.83122**. We analyze what techniques proved effective and discuss the challenges encountered. Code and model weights are available at [GitHub](#), [Model](#).

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: We fine-tune the LLaMA-3 8B model using Low-Rank Adaptation (LoRA) to determine whether a student’s mathematical answer is correct. Our approach combines efficient LoRA tuning with structured prompt design and incremental data expansion to improve reasoning consistency and accuracy.

Contributions:

- Efficient LoRA based fine tuning: Demonstrated that rank $r = 8$ with $\alpha = 16$ achieves 0.83 test accuracy on 4-bits quantization model
- Prompt and data scaling effects: Showed that structured prompts and incremental dataset growth significantly stabilize convergence and improve reasoning reliability.

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

Dataset size: 60k training, 5k validation, 10k test samples

Label distribution: 60.01% correct, 39.99% incorrect

Average lengths: Questions (64.8 tokens), Answers (4.4 tokens), Solution (170.7 tokens)

2.3 Preprocessing

We performed minimal preprocessing, including whitespace normalization and prompt construction. Each instance combines the question, answer, and optionally the student’s reasoning solution into a single textual input. Incorporating the solution text provides richer context for the model to assess correctness. We construct prompts by combining question, answer, and optionally solution text following the template 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 20M parameters.

Our LoRA configuration: rank $r = red8$, alpha $\alpha = 16$, dropout = $red0$, targeting attention (query/key/value/output) and feed forward (gate/up/down) projection matrices.

To fit the 8B parameter model into limited GPU memory, we adopt 4-bit quantization via the bitsandbytes backend. During fine tuning, only the low rank adapter weights are trained in FP16 precision, consistent with the QLoRA [3] paradigm.

3.2 Prompt Engineering

We formulate the task as an instruction-following problem. Our final prompt template:

```
You are a great mathematician and you are tasked with finding if a solution to a given maths question is correct or not. Your response should be 'True' if the solution is correct, otherwise 'False'. Below is the Question and Solution.  
Question: {question}  
GivenAnswer: {answer}  
GivenSolution (optional reasoning): {solution}  
Output:
```

We include the full solution text in the prompt to enrich contextual information and guide the model’s reasoning. Unlike humans, LLMs do not perform explicit logical deduction; instead, they predict the most likely continuation of the given context. When only the final answer is provided, the model may misjudge correctness due to internal computation or reasoning errors.

By adding the detailed solution field, the model can leverage linguistic cues and logical descriptions to

infer whether the reasoning process and final result are coherent.

In effect, the model learns to assess whether the solution itself is logically consistent and whether its conclusion aligns with the given answer.

3.3 Training Configuration

Table 1 shows our training hyperparameters. We fine-tuned Llama-3 8B using LoRA with a samll learning rate schedule to ensure stable convergence and prevent catastrophic forgetting.

Hyperparameter	Value
Learning Rate	2e-4 (final stage 5e-5)
Batch Size	4
Gradient Accumulation	8
Epochs	3
Max Sequence Length	2048
Warmup ratio	0.03
Weight Decay	0.01
Optimizer	AdamW
LR Scheduler	linear

Table 1: Training hyperparameters

We adopted a moderate base learning rate (2e-4) to accelerate early convergence and reduced it to 5e-5 in the final stage to stabilize fine-tuning as the model approached convergence.

Each incremental fine-tuning phase trained for 3 epochs using cross-entropy loss.

Training was performed on an NVIDIA A100 40GB GPU, with the complete multi-stage fine-tuning taking approximately 6 hours in total.

4 Experiments and Results

4.1 Baseline Performance

We first evaluate the provided baseline model, which achieves 0.726 test accuracy. The initial setting insert LoRA adapters with rank $r=1$ ($\alpha = 2$, dropout=0) into the attention projections q, k, v, o and MLP blocks gate, up, down, keeping all base weights frozen.

4.2 Experimental Setup

We perform a series of experiments to evaluate how LoRA configuration and learning rate affect model performance. Specifically, we design a **grid search**

over LoRA rank r , scaling factor α , dropout rate, and learning rate, as summarized in Table 2. Each configuration is trained for 100 steps on the same 5k subset of the training data to enable rapid grid search and identify the most promising hyperparameter combinations before scaling up full training. We then evaluate accuracy on the 500-sample validation set using the same inference prompt. The top two configurations are selected for full-scale fine-tuning on the complete dataset.

Hyperparameter Grid	Values
LoRA rank (r)	[2, 4, 8]
LoRA α	[4, 8, 16, 32]
LoRA dropout	[0.0, 0.05]
Learning rate	[$1e^{-4}$, $2e^{-4}$]
Max steps	100

Table 2: Hyperparameter search space for ablation experiments.

4.3 Results

Table 3 summarizes our experimental results. Starting from a baseline accuracy of 0.726, we conducted grid based experiments to study the effects of LoRA rank r , scaling factor α , dropout, and learning rate.

LoRA r	Dropout	LR	Steps	α	Val Acc.
2	0.0	$1e^{-4}$	100	4	0.720
2	0.0	$2e^{-4}$	100	4	0.696
2	0.05	$1e^{-4}$	100	4	0.718
2	0.05	$2e^{-4}$	100	4	0.698
4	0.0	$1e^{-4}$	100	8	0.738
4	0.0	$2e^{-4}$	100	8	0.706
4	0.05	$1e^{-4}$	100	8	0.734
4	0.05	$2e^{-4}$	100	8	0.718
8	0.0	$1e^{-4}$	100	16	0.718
8	0.0	$2e^{-4}$	100	16	0.736
8	0.05	$1e^{-4}$	100	16	0.708
8	0.05	$2e^{-4}$	100	16	0.748

Table 3: Grid search over LoRA configuration and learning rate. Higher α and moderate dropout yield consistently better validation accuracy.

Overall, the grid search reveals clear trends in how LoRA capacity and scaling affect model performance. Increasing the rank from 2 to 8 improves validation accuracy by approximately **+2-3%**, indicating

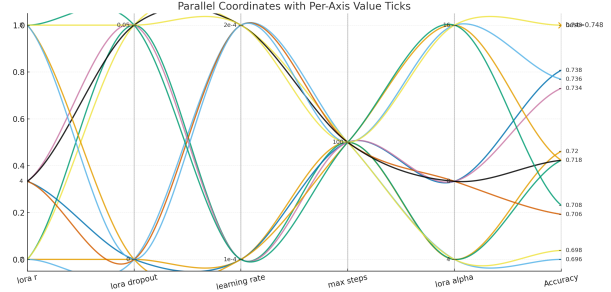


Figure 1: Parallel coordinate

that larger low rank subspaces enable more expressive adaptation under 4 bit quantization. Similarly, doubling the scaling factor α from 8 to 16 yields an additional **+1%** gain on average, suggesting that stronger adapter reweighting helps the model better align representations with the classification mission. The effect of dropout is modest but consistent, introducing a small dropout (0.05) slightly improves generalization (+0.5%) by mitigating overfitting during the short 100 step search.

We visualize the validation accuracies across all configurations in Figure 1. Our results show that **ranks** ($r=8$) combined with **scaling factors** ($\alpha=16$) yield the best performance. These settings were used for our final model.

Based on the grid search results, we selected the configuration with **LoRA rank $r = 8$, scaling factor $\alpha = 16$, dropout = 0.05, and learning rate = $2e^{-4}$** for the full scale training. We then performed incremental fine tuning in four stages - on the 0-5k, 5k-30k, 30k-40k, and 40k-60k subsets - to progressively expand data coverage and observe convergence behavior. The model exhibited steady improvement in validation accuracy up to 40k samples, beyond which the accuracy plateaued even after reducing the learning rate to $5e^{-5}$. This indicates that most reasoning patterns were already captured by the 40k subset, and additional data contributed limited new reasoning diversity.

4.4 What Worked

- **Incorporating the solution text.** Adding the step by step solution text improved reasoning consistency, learning to a increase in a validation accuracy compared to using only the question-answer pair. The model leveraged linguistic cues in the reasoning to decide whether

an answer was correct.

- **LoRA configuration tuning.** LoRA rank ($r = 8$) with doubled α values ($\alpha = 16$) and a dropout of 0.05 achieved the best trade off between capacity and generalization. The best model ($r=8, \alpha=16$) reached 0.748 accuracy on validation set.
- **Incremental fine tuning with lower learning rate.** Reducing the learning rate to 5e-5 for the final full epoch fine tuning stabilized the training curve and mitigated overfitting.

4.5 What Didn't Work

- **Data scaling beyond 40k samples.** When the training data size increased from 5k to 40k, test accuracy steadily improved, reaching approximately **0.83**. However, adding another 20k samples (totaling 60k) did not yield further gains. This suggests that the model had already captured the key reasoning patterns from the existing examples, and that additional samples introduced redundant information instead of new reasoning diversity.

4.6 Error Analysis

Example Error:

Question: How many positive three-digit integers less than 500 have at least two digits that are the same?

Solution: To calculate the answer, we need to enumerate all the positive three-digit integers less than 500, and select those who have at least two identical digits. Here is some Python code to calculate this. `<llm-code> count = 0 for number in range(100, 500): num_str = str(number) digits = set() for digit in num_str: if digit in digits: count += 1 break else: digits.add(digit)`

`print(count) </llm-code> <llm-code-output> 112 </llm-code-output>` Hence, the answer is 112.

Ground Truth: True *Model Prediction:* False

Analysis:

The model misclassified this sample mainly because the `solution` field included raw code and execution results (e.g., `<llm-code>` and `<llm-code-output>` blocks). Although the code was logically correct, the model lacks the ability to execute or verify code outputs. Instead, it likely

relied on textual cues and formatting patterns, leading it to misjudge the correctness of the reasoning. This shows that the model may overemphasize surface patterns (e.g.: presence of code or boxed results) rather than performing consistent logical validation. Removing or summarizing code blocks before inference could reduce such errors.

5 Discussion

Our final model achieves 0.83122 accuracy, representing a 10.522% improvement relative to the baseline. The improvement primarily comes from two factors: (1) incorporating the detailed solution text in the training prompt, which provides richer reasoning context for the model to assess correctness, and (2) performing systematic hyperparameter tuning over LoRA rank, scaling factor, and learning rate. Despite using a LoRA rank ($r = 8$) and 4-bit quantization, the fine tuned model preserves strong reasoning capability while keeping training efficient and memory light. This demonstrates that parameter efficient fine tuning can effectively adapt large models for logical verification tasks without full weight updates.

Challenges: Despite the effectiveness of incremental fine tuning, we observed diminishing returns beyond 40k samples. Even after adjusting the learning rate schedule, performance improvements were marginal. We hypothesize that this plateau stems partly from the capacity limitation of the 4 bit quantized base model and the restricted expressiveness of the prompt template, which may prevent the model from internalizing more diverse reasoning structures.

6 Conclusion

We successfully fine-tuned Llama-3 8B for math answer verification using LoRA-based parameter-efficient training. Through prompt refinement and systematic hyperparameter exploration, the model achieved strong performance while maintaining high computational efficiency. Our findings highlight that reasoning oriented fine tuning benefits more from well structured context than from simply scaling parameter size.

Future Work: Future improvements may include integrating lightweight programmatic verifiers to check arithmetic consistency, exploring ensemble scoring over multiple prompt variants, and scaling to

longer context models for handling multi step symbolic reasoning. Besides, it is worthy to explore higher bit quantization to further enhance reasoning generalization.

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, Yanzhi 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] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient Finetuning of Quantized LLMs. In *Proceedings of the 37th Conference on Neural Information Processing Systems (NeurIPS)*, 2023. arXiv:2305.14314.

Acknowledgments

Some figures, tables, and phrasing in this report were assisted by AI tools (chatGPT) to improve clarity and visual presentation.