Midterm Kaggle Competition

Naman Limani , Naman Vashishta
New York University
Deep Learning - Fall 2025
nl2833@nyu.edu, nv2375@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 significant improvement over the baseline, raising accuracy from 0.46 to 0.833, achieving a final test accuracy of **0.833**. We analyze what techniques proved effective and discuss the challenges encountered. Best model weights are available at https://github.com/NamanVashishta/deep-learning-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 , 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) to make fine-tuning tractable while maintaining performance.

Our Approach: Our approach centers on correct prompt engineering for the Llama-3 8B model. We identified that providing the `Expected Answer` as explicit context, in addition to the `Question` and `Solution`, was critical. We combine this prompt strategy with LoRA-based fine-tuning on a 50,000-sample subset of the data.

Contributions:

- Identified that the task is impossible without providing the `Expected Answer` in the prompt, a fix that improved accuracy from 0.46 to over 0.80.
- Determined that a 50k sample dataset provided the best performance, striking a balance between data diversity and training time.
- Found that a shorter sequence length (1024) significantly outperforms longer ones (4096), suggesting longer contexts add noise and dilute attention.

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

For our best-performing model, we used a subset of the available data.

Dataset size: 45,000 training, 5,000 validation (from a 50k subset), and 10,000 test samples.

Test Set Label Distribution (from model predictions): Our final model's predictions on the 10,000-sample test set resulted in 36.1% `True` (3,611) and 63.9% `False` (6,389) labels.

2.3 Preprocessing

Our primary preprocessing step was formatting the raw data into the Llama 3 chat template. We construct a prompt by combining the 'question', 'answer' (as 'Expected Answer'), and 'solution' text, along with the required system and user tags, as shown in Section 3.2.

3 Methodology

3.1 Model Architecture

Llama-3 8B: We use Llama-3 8B as our base model unsloth/meta-llama-3.1-8b-unsloth-bnb-4bit), a state-of-the-art transformer with 8 billion parameters.

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 42M parameters.

Our LoRA configuration (from `adapter_config.json`): rank $r = 16$, alpha $\alpha = 32$, dropout $= 0.0$, targeting all linear projection matrices (`q_proj`, `k_proj`, `v_proj`, `o_proj`, `gate_proj`, `up_proj`, `down_proj`).

We applied 4-bit quantization (as part of the `unsloth` base model) using bitsandbytes to reduce the memory footprint.

3.2 Prompt Engineering

We formulate the task as an instruction-following problem. After initial failures, we identified that the model must be provided with the 'Expected Answer' to perform verification.

Our final, successful prompt template uses the Llama 3 chat format:

```
<|begin_of_text|>
<|start_header_id|>
system<|end_header_id|>
You are a meticulous math solution
verifier.Respond ONLY with 'True' or
'False'.
<|eot_id|>
<|start_header_id|>
user<|end_header_id|>
Question:
{question}
Expected Answer:
{answer}
Provided Solution:
{solution}
<|eot_id|>
<|start_header_id|>assistant
<|end_header_id|>
```

Hyperparameter	Value
Learning Rate	1e-4
Batch Size (per device)	4
Gradient Accumulation	2
Epochs	1
Max Sequence Length	1024
Warmup Steps	100
Weight Decay	0.01
Optimizer	AdamW (8-bit)

Table 1: Training hyperparameters for the final model.

Initial experiments with a prompt missing the `Expected Answer` field failed, yielding only 0.46 accuracy as the model attempted to solve the problem rather than verify it. Adding the `Expected Answer` as explicit context was the single most important change, enabling the model to perform the correct verification task.

3.3 Training Configuration

Table 1 shows our final model's hyperparameters. We used a standard `1e-4` learning rate for LoRA and a batch size of 4 with 2 gradient accumulation steps, resulting in an effective batch size of 16 on our dual-GPU setup.

We use cross-entropy loss and train for 1 epoch on dual T4 GPUs. Training takes approximately 8.25 hours.

4 Experiments and Results

4.1 Baseline Performance

The provided competition baseline achieved 0.726 test accuracy. Our initial model, which used a logically flawed prompt, performed even worse, scoring 0.46. This confirmed that the model was not performing the correct task.

4.2 Experimental Setup

We conducted ablation studies to find the optimal configuration. **Experiment 1: Prompt Engineer-**

Configuration	Data	Val	Test
Baseline	50k	0.60	0.726
Failed Prompt	50k	0.45	0.460
Exp 1 (60k, 4096sl)	60k	0.63	0.82
Exp 2 (30k, 2048sl)	30k	0.59	0.80
Exp 3 (50k, 1024sl)	50k	0.65	0.83
Final Model	**50k**	**0.650**	**0.833**

Table 2: Experimental results comparison. Our final model used 50k samples and 1024 sequence length.

ing: Our first experiment was to fix the prompt format to include the 'Expected Answer' (as shown in Sec 3.2) and establish a new, functional baseline.

Experiment 2: Data Sizing: Using the correct prompt, we tested training on 30k, 50k, and 60k data samples to measure the impact of data volume.

Experiment 3: Sequence Length: We compared sequence lengths of 1024, 2048, and 4096 to find the optimal context.

4.3 Results

Table 2 summarizes our key experimental results. Our results show a massive 37% improvement from fixing the prompt (0.460 to 0.833). We also see that 50k samples at 1024 sequence length was the optimal configuration.

Notably, all local validation scores were unreliable and did not correlate with test performance, suggesting the 10% validation split was not representative of the test set.

Our final model, which achieved 0.833 accuracy on the 10,000 test samples, predicted 3,611 (36.1%) samples as `True` and 6,389 (6.39%) samples as `False`. This suggests the test set is skewed towards incorrect solutions.

4.4 What Worked

- **Correct Prompt Engineering:** The jump from 0.46 to 0.83+ was achieved by providing the `Expected Answer` to the model, which was the key insight.
- **Optimal Data Size:** 50,000 samples proved to be the sweet spot, outperforming 30k (0.80). This gave the model enough data to learn the "trick" verification task.
- **Shorter Sequence Length:** Our best model (0.833) used a 1024 sequence length. Our 4096-length model (0.820) performed worse even with more data (60k), indicating that longer contexts added noise.

4.5 What Didn't Work

- **The Initial Prompt:** Our first model failed completely (0.46) because the prompt was logically flawed and did not provide the model with the necessary information to complete its task.
- **Longer Sequence Lengths:** We hypothesized that `max_seq_length` of 2048 or 4096 would be better, but they consistently underperformed our 1024 model.
- **Relying on Validation Score:** Our validation scores (0.59-0.65) were misleading and showed no correlation with test accuracy. Trusting this score would have led to poor model choices.

4.6 Error Analysis

Our primary failure case, identified through debugging, was the model's tendency to revert to a "solver" (e.g., "Let's use sympy...") when the prompt was ambiguous.

Example Error Class: The dataset contains "trick" questions where the `Provided Solution` has flawed reasoning (e.g., using wrong variables) but coincidentally arrives at the correct `Expected Answer`. Our final model, even at 0.833 accuracy, can still be fooled by this "correct answer" and misclassify the reasoning as `True`.

5 Discussion

Our final model achieves **0.833** accuracy, representing a 10.7% improvement relative to the 0.726 baseline. Our approach worked because it was centered on fixing the core logic of the prompt. By providing all necessary information ('Question', 'Expected Answer', 'Solution'), we successfully shifted the model's behavior from a "solver" to a "verifier".

Challenges: The primary challenge was identifying the non-representative validation split.

The low validation scores (0.59-0.65) falsely suggested our models were failing, when in fact they were performing well on the test set. Overcoming this required trusting our one-epoch test submissions over our local validation metrics, which proved to be the correct strategy.

6 Conclusion

We successfully fine-tuned Llama-3 8B for math answer verification using LoRA-based parameter-efficient fine-tuning, a 50k dataset, and a 1024 sequence length, achieving a 0.833 test accuracy. Our key findings are that task success is critically dependent on a logically complete prompt (including the `Expected Answer`) and that shorter sequence lengths are optimal for this task.

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.

Acknowledgments

We utilized Gemini for debugging assistance and comment generation.