DL-Fall-25 Kaggle Contest Report

Lejun Zhang
New York University
lz3247@nyu.edu

Lingfei Zhu
New York University
lz3408@nyu.edu

Ziyuan Hong
New York University
zh3199@nyu.edu

Abstract

This work investigate math answer verification as a binary classification problem and adapt instruction tuned decoder only LLMs with parameter efficient fine tuning. Using LLaMA 3 8B, we compare LoRA and QLoRA under two objectives, constrained generative labeling with `True`/`False` and a lightweight classification head, and under two input settings, question plus answer and question plus answer plus solution. On a competition dataset, QLoRA with rank 16 and scale 32, together with cosine decay and gradient clipping, achieves 0.81 accuracy and exceeds the baseline. Providing worked solutions consistently improves performance, and constrained decoding leverages the pretrained language head to yield strong calibration, while the classification head is competitive with mild label smoothing. Ablations show that explicit instruction prompts stabilize training, whereas overly long contexts and excessive adapter dropout degrade accuracy. The code is available at https://github.com/zhanglejun02/DL-Fall-25-Kaggle-Contest

1 Introduction

Determining whether a proposed answer to a mathematical question is correct is a fundamental capability for educational assistants and automated graders. The task requires the model to connect the problem statement, the candidate answer, and an optional solution explanation, and to decide a binary label that indicates correctness. We address this problem by supervised finetuning a modern large language model on a curated competition dataset for math question answer verification. We adopt LLaMA3-8B as the base model, and we apply parameter efficient finetuning with Low Rank Adaptation in order to reduce memory footprint and to enable stable training on limited hardware. Our goal is to assess whether instruction tuned generative checkpoints can be adapted into accurate verifiers, and to analyze design choices that impact performance, such as the inclusion of solution text, the choice of prompting and decoding, and the loss formulation.

2 Related Work

Low-rank in Weight. LoRA shows the potential of memory-efficient learning for LLM with the intrinsic low-rank structure (Hu et al., 2022). (Schotthöfer et al., 2022) constrains the rank of weight matrices to find "winning-ticket" dynamically in dense network. ReLoRA (Lialin et al., 2023) utilizes locally low-rank updates to train high-rank networks. RandLoRA (Albert et al., 2025) updates a learned linear combinations of low-rank, non-trainable random matrices. Dora (Liu et al., 2024) decomposes weights to directional matrix and magnitude vector for fine-tuning. Further works improve the low-rank adapter with the help of SVD to the weights in the initialization of the adapters (Meng et al., 2024), only training the top-r singular values (Sun et al., 2024), or the corresponding coefficients (Lingam et al., 2024). Lora-Null (Tang et al., 2025) finds the null space of representative activations and initializes the LoRA adapter with the pre-trained weights in that null space. (Wang et al., 2024) achieves high-rank updates with low costs by selecting skeletons from the pre-trained weight and learning a small matrix instead. (Jaiswal et al., 2024) unifies weight compression and memory-efficient fine-tuning as one to do adaptive low-rank weight projection.

Supervised Fine Tuning. Supervised fine-tuning (SFT) aligns pretrained language models to task instructions by optimizing on paired *instruction–response* data, which markedly improves zero-shot generalization and controllability. Early work demonstrated that instruction-tuned models become strong zero-shot learners when exposed to diverse task templates (Wei et al., 2021). In practical alignment pipelines, SFT is the first stage

before preference-based optimization, where human demonstrations are used to shape model behavior toward helpful and safe responses (Ouyang et al., 2022). To reduce reliance on costly human curation, *self-instruction* bootstraps synthetic instruction–response pairs from a seed set, then filters and re-trains to expand coverage efficiently (Wang et al., 2022). Data quality is a key driver of SFT effectiveness. With carefully curated exemplars, even limited SFT data can yield competitive assistant-style behavior, suggesting that most knowledge is inherited from pretraining while SFT mainly teaches formatting and intent following (Zhou et al., 2023). The efficiency of SFT has advanced through parameter-efficient fine-tuning, notably LoRA, which inserts low-rank adapters and trains a small number of parameters while freezing the base model (Hu et al., 2022). Quantization-aware methods such as QLoRA further enable memory-frugal SFT of large models by combining 4-bit quantization with low-rank adapters and paged optimizers without significant quality loss. These developments make SFT practical on commodity hardware and have become standard for domain adaptation and instruction following at scale.

3 Dataset

This study uses the Math Question Answer Verification Competition dataset. Each example comprises four fields: a natural language *question*, a scalar *answer* representing the ideal target, an optional *solution* that provides step by step reasoning, and a boolean label *is_correct*. The training split contains pairs whose labels indicate whether the provided answer matches the ground truth under the rules of the problem domain. The test split provides the same inputs, and its labels are placeholders set to `True`, with leaderboard evaluation based on model predictions. No manual cleaning is performed. Duplicate rows are removed by hashing the triple of *question*, *answer*, and *solution*. Text is not lowercased. Mathematical symbols are preserved as provided.

Preprocessing We format each example into a single packed sequence. Numbers are kept as raw text. We do not apply special numeric tokenization. We truncate only when the total length exceeds the model context window. We study two input variants, *QA* which concatenates question and answer, and *QAS* which concatenates question, answer, and solution.

4 Model Description

4.1 LLaMA Family of Models

LLaMA is a family of decoder only transformer language models that provide strong language modeling capability with efficient training and inference footprints (Touvron et al., 2023a,b). The architecture follows the standard causal transformer with rotary position embeddings, multi head self attention, and feed forward blocks with gated activation. LLaMA adopts a byte pair encoding tokenizer and is trained on a large and diverse corpus of public text. The design objective is to obtain competitive performance under a transparent research license and to enable downstream adaptation through lightweight fine tuning methods. Compared with earlier open models, LLaMA emphasizes careful data curation, stable optimization, and well tuned scaling that together yield favorable perplexity and robust transfer to instruction following after supervised or reinforcement learning based alignment. In typical research practice, LLaMA checkpoints serve as the base model for domain adaptation, instruction tuning, or task specific supervised fine tuning.

4.2 Low Rank Adaptation of Large Models

LoRA is a parameter efficient fine tuning method that injects trainable low rank matrices into the linear projections of a pretrained transformer (Hu et al., 2022). Let $W_0 \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ denote a frozen weight matrix. LoRA learns a low rank update $\Delta W = BA$ with $A \in \mathbb{R}^{r \times d_{\text{in}}}$ and $B \in \mathbb{R}^{d_{\text{out}} \times r}$ where $r \ll \min(d_{\text{in}}, d_{\text{out}})$. The adapted layer computes

$$h(x) = W_0 x + \alpha BAx,$$

where α is a scaling factor that stabilizes training. Only A and B are updated while W_0 remains frozen. This factorization reduces trainable parameters and optimizer state which lowers memory usage and speeds up fine tuning. LoRA is commonly applied to the query and value projections in attention and optionally to feed forward layers. Typical hyperparameters include the rank r in the range from four to sixteen, the scaling α matched to r, and a small dropout on the LoRA branch to improve generalization. The method composes linearly with other efficiency techniques and preserves the original inference path because the low rank update can be merged into W_0 after training.

Quantized LoRA QLoRA extends LoRA by performing fine tuning on an NF4 or INT4 quantized copy of the base model while keeping the LoRA adapters in higher precision (Dettmers et al., 2023). The quantization reduces activation and weight memory which enables fine tuning of larger base models on commodity accelerators. Empirical results show that QLoRA recovers most of the full precision performance with substantially lower memory requirements.

Practical guidance For LLaMA based supervised fine tuning, LoRA and QLoRA provide strong cost effectiveness. A recommended baseline uses rank eight adapters on attention projections, scaling $\alpha = 16$, adapter dropout between zero and five percent, and AdamW with a learning rate in the range 2×10^{-4} to 5×10^{-4}. Training stability benefits from cosine or linear decay with a small warmup ratio and from gradient clipping. Checkpoints can be stored as standalone LoRA weights which simplifies deployment and model sharing.

LLaMA offers a strong and accessible base model for language understanding and generation. LoRA and its quantized variant enable efficient specialization to new tasks with minimal computational overhead. The combination supports rapid iteration, reproducibility, and deployment in academic and industrial settings.

4.3 Formulating Binary Verification

We consider two training objectives.

Generative label tokens. We keep the language modeling head and train the model to generate a short textual verdict that maps to a binary label. We use a constrained target string, either `True` or `False`. Let $y \in \{0, 1\}$ denote the target, and let $t(y)$ denote the corresponding token sequence. The loss is the negative log likelihood of $t(y)$ given the prompt.

Classification head. We attach a linear classifier on the final hidden state of a sentinel position. Let $h \in \mathbb{R}^d$ denote the hidden state at the first generated position, the classifier outputs $p(y = 1 \mid x) = \sigma(w^\top h + b)$. The loss is binary cross entropy. We compare both variants in Section 6.

5 Experimentation and Hyperparameter Settings

5.1 Prompt Template

We found that an explicit instruction improves stability. The following template is used for the generative objective.

```
You are a strict math answer verifier.
Decide if the provided answer is correct for the ques.
Respond with exactly one token: True or False.

[Question]
{question}

[Answer]
{answer}

[Solution]
{solution_or_empty}

Verdict:
```

For the QA variant we omit the Solution block. For the classification head we use the same prompt and read the hidden state at the first position after `Verdict:`.

5.2 Training Details

We fine tune only the adapter parameters using AdamW with decoupled weight decay, while keeping the base model frozen in four bit `NF4` and storing adapter weights in `bfloat16`. The learning rate follows a cosine decay schedule with a linear warmup ratio of three percent, and we apply gradient clipping by global norm. To reach the desired global batch size we employ gradient accumulation and set the effective global batch to 128 sequences. The LoRA configuration uses rank $r = 16$, scaling $\alpha = 32$, and dropout 0.05, and we target both attention projection modules and feed forward projections. The maximum training sequence length is $L_{\text{train}} = 1024$. For optimization we use AdamW with $\beta_1 = 0.9$, $\beta_2 = 0.95$, and weight decay 0.05, and we select the learning rate on the validation set from $\{1.5 \times 10^{-4}, 1.0 \times 10^{-4}, 5.0 \times 10^{-5}\}$. Training runs from one to three epochs with early stopping on validation loss. For the objective we use negative log likelihood for the generative variant and binary cross entropy for the classification variant, and we apply label smoothing of 0.05 only to the classification head while leaving the generative targets unsmoothed.

5.3 Data Splits and Evaluation Protocol

Stratified partitions of the original training set produce training, validation, and holdout subsets whose ratios sum to one, with class balance preserved across splits. Accuracy serves as the primary evaluation metric. Additional metrics include F1 score, precision, recall, area under the receiver operating characteristic curve, and expected calibration

error. Model selection uses the checkpoint that attains the highest validation accuracy. Final results are reported once on the holdout set. For the generative variant, logits of the tokens forming `True` and `False` are converted to probabilities with length normalization. For the classification–head variant, a default decision threshold of 0.5 is applied unless calibration on the validation set indicates a different value. When noted, temperature scaling is fitted on the validation set to improve calibration.

5.4 Ablations

We perform ablations over input composition and objective.

- QA versus QAS inputs
- Generative labels versus classification head
- With and without instruction preface
- Short context, length $L = 512$, versus long context, length $L = 2048$

6 Results

The final accuracy reached 0.81, successfully surpassing the baseline. The QAS input improves accuracy compared to QA, which shows that the solution text is informative for verification. The generative objective outperforms the classification head when we enforce a strict two token vocabulary, which suggests that the pretrained language head aligns well with the binary verdict space. The classification head benefits from label smoothing and from temperature scaling, which reduces overconfidence and improves F1.

What worked Including the solution explanation consistently improved accuracy. A short instruction before the inputs stabilized training and reduced degenerate outputs. Constraining the generative decoder to the tokens that form `True` and `False` improved both accuracy and calibration. QLoRA with rank sixteen and scaling thirty two delivered the best trade off between speed and quality under our memory budget.

What did not work Training a large classification head on top of pooled token features degraded performance, which indicates a mismatch with the pretrained representation geometry. Removing the instruction header increased the rate of verbose outputs that did not map to the label space, which harmed accuracy. Very long contexts beyond the training maximum did not help, and introduced instability for small batch sizes. Over aggressive dropout on LoRA adapters reduced accuracy.

Error Analysis Most false positives occur when the numeric answer matches by coincidence but the reasoning contradicts constraints in the question. Most false negatives occur when the correct answer is formatted with additional units or a boxed notation that the model interprets as mismatched. Normalizing number formats reduces such errors. Future work can incorporate a lightweight numeric parser in the prompt to mitigate formatting variance.

7 Conclusion

In this work we presented a parameter efficient approach to math answer verification by finetuning LLaMA with LoRA. Generative labeling with a constrained verdict vocabulary is a strong baseline, and it benefits from including solution explanations in the input. The classification head is competitive with proper regularization and calibration, but the generative objective remains simpler and robust. The findings suggest that pretrained language models can serve as verifiers with modest adaptation cost. Future directions include structured numeric normalization, retrieval of similar solved problems, and consistency training across multiple sampled rationales.

Reproducibility Checklist

We release prompt templates and training hyperparameters in this section. We define all random seeds for data split and parameter initialization. We fix the tokenizer version to the checkpoint default. We record the exact sequence length, batch size, and gradient accumulation steps. We log metrics with a fixed evaluation frequency in tokens learned.

References

Paul Albert, Frederic Z Zhang, Hemanth Saratchandran, Cristian Rodriguez-Opazo, Anton van den Hengel, and Ehsan Abbasnejad. 2025. Randlora: Full-rank parameter-efficient fine-tuning of large models. *arXiv preprint arXiv:2502.00987*.

Tim Dettmers, Mike Lewis, Sam Shleifer, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms. *arXiv preprint arXiv:2305.14314*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang,

Weizhu Chen, et al. 2022. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3.

Ajay Jaiswal, Lu Yin, Zhenyu Zhang, Shiwei Liu, Jiawei Zhao, Yuandong Tian, and Zhangyang Wang. 2024. From galore to welore: How low-rank weights non-uniformly emerge from low-rank gradients. *arXiv preprint arXiv:2407.11239*.

Vladislav Lialin, Namrata Shivagunde, Sherin Muckatira, and Anna Rumshisky. 2023. Relora: High-rank training through low-rank updates. *arXiv preprint arXiv:2307.05695*.

Vijay Chandra Lingam, Atula Neerkaje, Aditya Vavre, Aneesh Shetty, Gautham Krishna Gudur, Joydeep Ghosh, Eunsol Choi, Alex Dimakis, Aleksandar Bojchevski, and Sujay Sanghavi. 2024. Svft: Parameter-efficient fine-tuning with singular vectors. *Advances in Neural Information Processing Systems*, 37:41425–41446.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. 2024. Dora: Weight-decomposed low-rank adaptation. *arXiv preprint arXiv:2402.09353*.

Fanxu Meng, Zhaohui Wang, and Muhan Zhang. 2024. Pissa: Principal singular values and singular vectors adaptation of large language models. *Advances in Neural Information Processing Systems*, 37:121038–121072.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Steffen Schotthöfer, Emanuele Zangrando, Jonas Kusch, Gianluca Ceruti, and Francesco Tudisco. 2022. Low-rank lottery tickets: finding efficient low-rank neural networks via matrix differential equations. *Advances in Neural Information Processing Systems*, 35:20051–20063.

Chengwei Sun, Jiwei Wei, Yujia Wu, Yiming Shi, Shiyuan He, Zeyu Ma, Ning Xie, and Yang Yang. 2024. Svfit: Parameter-efficient fine-tuning of large pre-trained models using singular values. *arXiv preprint arXiv:2409.05926*.

Pengwei Tang, Yong Liu, Dongjie Zhang, Xing Wu, and Debing Zhang. 2025. Lora-null: Low-rank adaptation via null space for large language models. *arXiv preprint arXiv:2503.02659*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, et al. 2023a. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Hugo Touvron, Louis Martin, Kevin Stone, et al. 2023b. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Qibin Wang, Xiaolin Hu, Weikai Xu, Wei Liu, Jian Luan, and Bin Wang. 2024. Pmss: Pretrained matrices skeleton selection for llm fine-tuning. *arXiv preprint arXiv:2409.16722*.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2022. Self-instruct: Aligning language models with self-generated instructions. *arXiv preprint arXiv:2212.10560*.

Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2021. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srinivasan Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, et al. 2023. Lima: Less is more for alignment. *Advances in Neural Information Processing Systems*, 36:55006–55021.