

# **Enhancing LLMs' Capabilities in Solving Math Problems: LLM-OPS GenAI Application**

Alexis Bryan Ambriz, Nick Kornienko, Joash Muganda, Kelly Nguyen

Charles Davidson School of Engineering San Jose State University

CMPE 297: Special Topics

Vijay Eranti

07 December 2024

# Abstract

In this study, we explore the enhancement of Large Language Models (LLMs) such as SmolLM2 1.7B and Llama 3.2 1B for solving mathematical integration, while expanding their capabilities to other mathematical operations like addition, derivatives, and roots. The project leverages a meticulous LLM-Ops pipeline approach, which guides the training, validation, and deployment phases of the model lifecycle, ensuring systematic and efficient model improvement. The initial phase of the project involves employing a pre-trained LLM as the foundation for subsequent fine-tuning tailored to specific mathematical problems. Using Python, we programmatically generate diverse datasets containing problems and solutions for integration, addition, derivatives, and roots. This approach not only furnishes the models with a wide range of training data but also circumvents the complexities and costs associated with using LLMs for dataset generation. For the fine-tuning process, the models are retrained on these generated datasets. We incorporate advanced training methodologies within the LLM-Ops framework to systematically enhance the models' mathematical problem-solving abilities. After training, the models undergo rigorous validation with separate datasets to objectively assess performance improvements, specifically evaluating the accuracy of solutions across different types of mathematical problems. To make these enhanced capabilities accessible, a user interface (UI) was developed using Hugging Face, allowing users to interact directly with the models and utilize them for solving mathematical problems in real-time. This UI not only demonstrates the practical application of the trained models but also facilitates further testing and user feedback in real-world scenarios. This research illustrates the potential of fine-tuned LLMs to revolutionize how mathematical problems are approached and solved in educational technology and beyond.

## Introduction

Large language models (LLMs) are adept at generating and understanding text but often falter with tasks requiring precise logical reasoning or numerical analysis, such as solving mathematical problems. Their inherent limitations in symbolic manipulation and interpreting math-related queries stem from a design that prioritizes linguistic over numerical understanding. Our project aims to refine these capabilities by implementing a comprehensive LLM-OPs pipeline to train and evaluate LLMs for responding to math problems with numerical and LaTeX outputs.

This enhancement addresses the critical need for accuracy in mathematical problem-solving within educational and professional settings. We have developed a method that integrates domain-specific datasets and fine-tuning techniques, significantly improving the LLMs' ability to process and output mathematical information correctly. Preliminary results indicate a substantial reduction in error rates and an increase in the precision of outputs, promising broader applications in scientific computing and education technology. This approach not only advances the functionality of LLMs in handling complex calculations but also ensures their outputs are practically useful in real-world scenarios.

## Related Work

In the context of enhancing LLMs for mathematical reasoning, several notable research efforts have been undertaken. The study "ChatGLM-Math: Improving Math Problem-Solving in Large Language Models with a Self-Critique Pipeline" presents a unique approach where a Self-Critique pipeline is employed. This method enhances both the linguistic and mathematical abilities of LLMs without favoring one over the other by integrating a Math-Critique model that evaluates the LLM's mathematical outputs and offers tailored feedback. This ensures the model's continual learning and alignment with mathematical accuracy [1].

Another significant work, "Large Language Models for Mathematical Reasoning: Progresses and Challenges," discusses various datasets and the implementation of models to address different types of mathematical problems, highlighting the ongoing efforts and challenges in refining LLMs' capabilities in this domain [2].

Compared to these studies, our approach involves using a full LLM-OPs pipeline for continuous evaluation and updating, alongside developing a user interface via Hugging Face. This not only advances the model's mathematical solving capabilities but also makes it accessible for practical use and real-world testing. Our method emphasizes creating domain-specific training datasets and leveraging fine-tuning techniques that align with real-world mathematical applications, potentially offering a more integrated solution for both academic and professional use.

# Data

We gathered addition, square root, and derivation based math problems. As we generated our own datasets using python code, the data was not heavily preprocessed as it was designed to meet the needs of our end goal: training the model to understand the context of each mathematical problem and be able to understand the logic. We followed the pattern of allocating ~80% of each dataset for training and the remaining for evaluation; The “additions” based subset of data is our largest and main dataset - it accounts for ~1,000,000 rows - whereas the remaining roots and derivatives based datasets account for 10,000 rows each.

## Addition

As the data is specific to the math domain, the data is inherently a combination of numerical and text. For example, the addition split of data uses the ‘+’ operator.

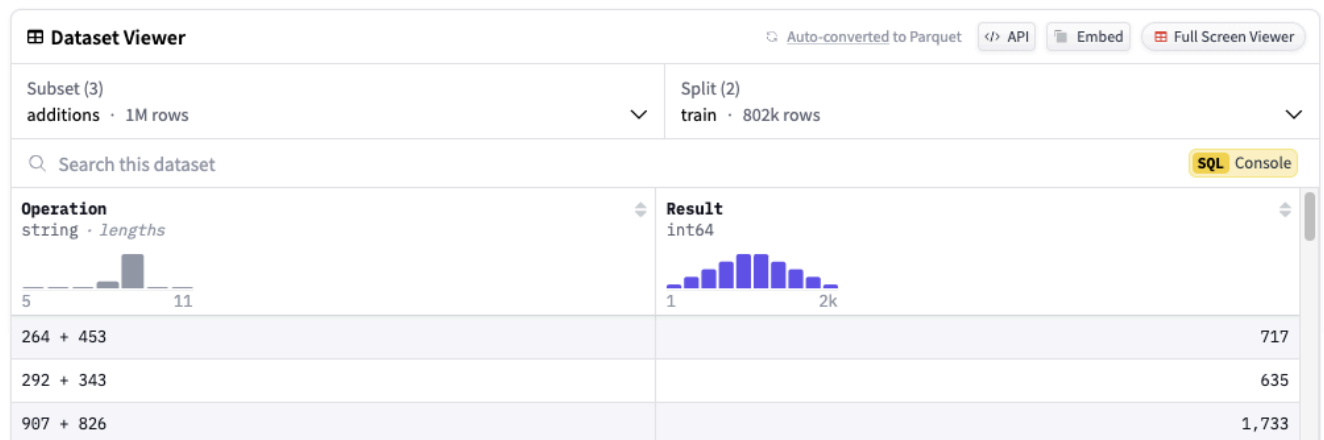


Fig 3. The Additions Subset

We incorporated a variety of math problems as it is likely that a language model that is trained to solve math problems for one type of problem (addition) can leverage the understanding it learned in adding numbers in other equations - such as in finding square roots and derivatives. The associative property, for example, states that “the sum or the product of three or more numbers does not change if they are grouped in a different way”.

# Roots

The derivative and roots based problems use LaTeX code to capture their own logical representation respectively. The data was generated with python code and is available within the publicly available repo that accompanies this paper.

Subset (3) roots · 10k rows	Split (1) train · 10k rows
Search this dataset	
Function string · lengths	Roots string · lengths
61~75 27.2%	1~242 49.1%
$\frac{1}{10} \left( -7x^2 - \frac{23}{4}x + \frac{63}{16} \right)^{19} x$	$-23/56 + \sqrt{6233}/168, -\sqrt{6233}/168 - 23/56$
$\frac{7}{\left( -\frac{3}{4}x^8 - \frac{4}{3}x^9 - x^2 - 2x \right)^2} x$	$-32/81 + 2484/(2187 \cdot (-1/2 - \sqrt{3}) \cdot (146251/19683 + \sqrt{391641}/81)^{1/3}) - 2 \cdot (-1/2 - \sqrt{3}) \cdot (146251/19683 + \sqrt{391641}/81)^{1/3}, -32/81 - 2 \cdot (-1/2 + \sqrt{3}) \cdot (146251/19683 + \sqrt{391641}/81)^{1/3}, -32/81 - 2 \cdot (-1/2 + \sqrt{3}) \cdot (146251/19683 + \sqrt{391641}/81)^{1/3}$
$\frac{2}{\left( -\frac{3}{13}x^3 - \frac{13}{2}x^{12} \right)^5} x^2$	$-13/18$

Fig 2. The Roots Subset

# Derivatives

However, having LaTeX formatted functions for the derivatives and roots based problems resulted in issues. The equations we used are complex. In hindsight, the model would require an external tool (that we did not add) to parse or correct the LaTeX generated by the LLM in order to evaluate the loss or calculate the root mean squared error (RMSE) on a validation set to return more accurate results. Previous research, such as with the “self-critique” method could potentially alleviate this by including an extra ‘evaluator’ model [1].

Subset (3) derivatives · 10k rows	Split (1) train · 10k rows
Search this dataset	
Function string · lengths	Derivative string · lengths
1 123	1 592
$49x^2 \sin(\left(3x\right)) \cos(\left(9x\right))$	$-441x^2 \sin(\left(3x\right)) \sin(\left(9x\right)) + 147x^2 \cos(\left(3x\right)) \cos(\left(9x\right)) + 98x \sin(\left(3x\right)) \cos(\left(9x\right))$
$300x^6$	$1800x^5$
$-\frac{x}{\log(\left(9x^2\right))}$	$-\frac{1}{\log(\left(9x^2\right))} + \frac{2}{\log(\left(9x^2\right))^2}$

Fig 3. The Derivatives Subset

# Methods

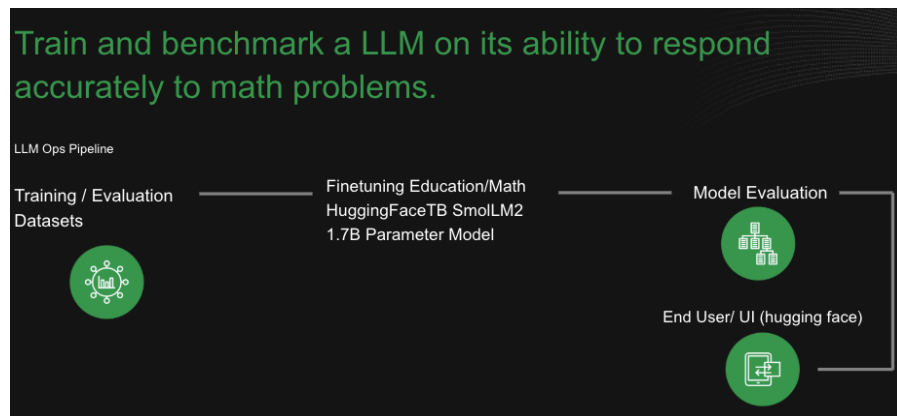


Fig 4. The End-to-End Pipeline

## LLaMA 1B Instruct

Our approach for creating a LLM trained on all subsets of the data started by first fine-tuning a small base LLaMA model of only 1 billion parameters. We utilized the Google Colaboratory cloud computing environment to standardize the build environment. Building the LLaMA 1B fine-tuned model helped set up a training pipeline for understanding how to load a quantized model, create a training prompt, train (fine-tune) the model using LoRa adapters on all subsets of the data, and then re-quantize and save the model to a HuggingFace model repo. We also utilized the Weights & Biases API during training to remotely save the training losses, the steps, and the learning rates.

## SmoLM2 1.7B Instruct

Loading a small base version of an LLM such as the LLaMA 1B Instruct model was beneficial in speeding up the training and development process. After setting up a pipeline with the LLaMA 1B model, it was simple to reproduce the process and fine tune a new model, the SmoLM2 1.7B Instruct model. We also used the popular fine tuning library Unsloth to load a 4 bit quantized version of the models and then re-quantize them; This lowered the memory requirements for training a model that we could then implement when using larger parameter models.

# Experiments and Results

## Training

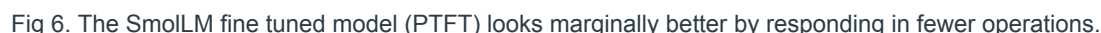
The roots and derivatives subsets were small and only trained for 500 steps each. The additions' dataset was used to train for 2000 steps as it was large. Furthermore, the additions' dataset was used to train the model before the model was fine-tuned on the roots and then derivatives subsets. In the top right visual is the learning rate for all three subsets. It shows how the learning rate increases linearly on the additions' training step prior to tapering off in the roots/derivatives steps.



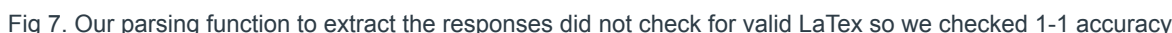
Fig 5. The Losses by Subset during Training of LLaMA 1B, and Learning Rates for all sets

While the loss in the above plots show that the model had lower initial and overall loss on the derivatives subset than on the roots subset, this is simply due to the derivatives subset being used to fine-tune after the roots subset was used to fine-tune. As all subsets of the data were used to fine-tune sequentially, the model was iteratively learning from one dataset prior to being used to learn another subset.

We attempted to evaluate the model using common evaluation metrics such as the mean squared error (MSE) or the root mean squared error (RMSE), that are typically applied to linear regression models that make numerical predictions. In the case of our models, we believe this would evaluate our model's ability to understand math, and not just language as is typical of other loss metrics for large language models. However, the data generating nature of our fine-tuned models resulted in outputs that strayed from the input prompt we used during training; Instead of outputting responses to the math queries in strictly numerical format, the models attempted to solve the input in multiple operations as text or responded in lengthy LaTeX text format.



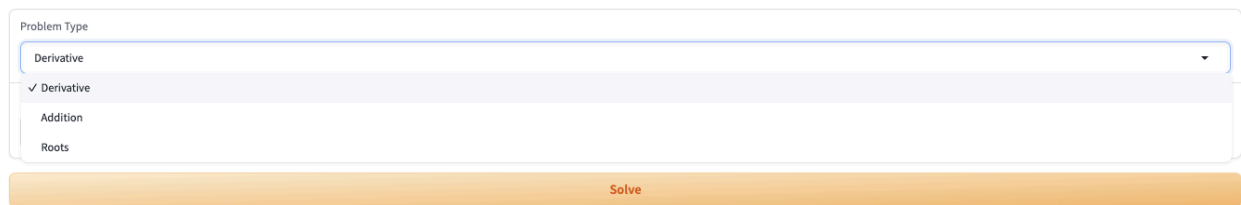
This made it difficult to process the responses in such a way that made it possible to calculate the MSE/RMSE or the accuracy percentage. We used a simple python regex function to attempt to parse the LaTeX or lengthy operations output to only output numerical values that could be used for calculating the evaluations metrics, however, this was unsuccessful.





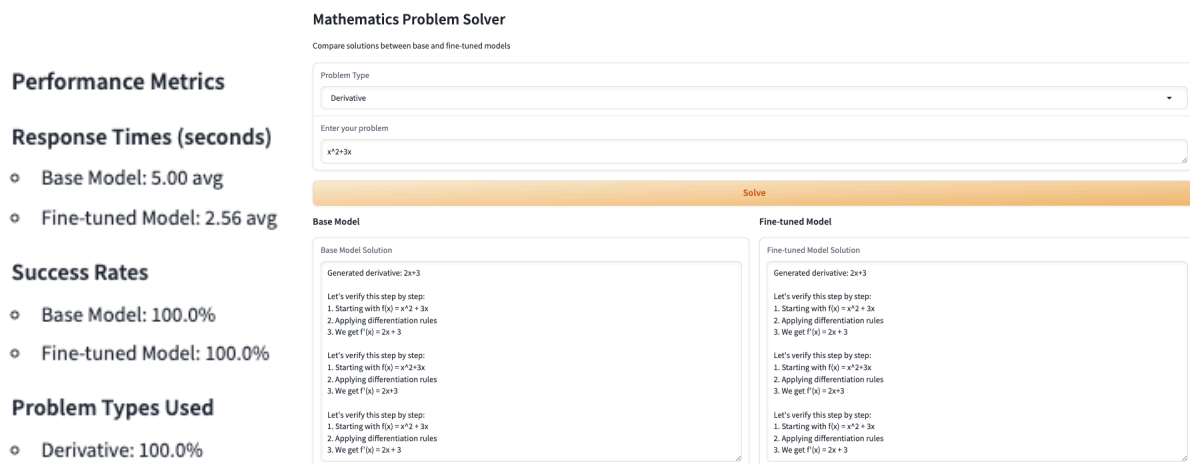
# User Interface

Our user interface was built using Gradio and was hosted on HuggingFace spaces. This was in-line with the cloud dataset, which was also stored in HuggingFace using their datasets class. The UI also includes a problem type drop-down menu that allows you to select the context for the problem that would reasonably be used for input prompting. It also includes tracking of run-time performance and logging metrics.



The image shows a web interface for selecting a problem type. At the top, there is a dropdown menu labeled "Problem Type" with "Derivative" selected. Below the dropdown, there is a list of options: "Derivative" (which is checked with a green checkmark), "Addition", and "Roots". At the bottom of the interface, there is a large orange button labeled "Solve".

Fig 8. The drop-down context menu allows specification based on the subsets of math data used for fine tuning.



The image shows a web interface titled "Mathematics Problem Solver". It compares solutions between a base model and a fine-tuned model. On the left, there are three sections: "Performance Metrics", "Response Times (seconds)", and "Success Rates". The "Performance Metrics" section shows "Base Model: 5.00 avg" and "Fine-tuned Model: 2.56 avg". The "Success Rates" section shows "Base Model: 100.0%" and "Fine-tuned Model: 100.0%". The "Problem Types Used" section shows "Derivative: 100.0%". In the center, there is a form with a "Problem Type" dropdown (set to "Derivative"), an "Enter your problem" input field (containing  $x^2+3x$ ), and a "Solve" button. Below the form, there are two columns: "Base Model" and "Fine-tuned Model". Each column shows the "Generated derivative:  $2x+3$ " and a step-by-step verification process. The "Base Model" solution is: "Let's verify this step by step: 1. Starting with  $f(x) = x^2 + 3x$ , 2. Applying differentiation rules, 3. We get  $f'(x) = 2x + 3$ ". The "Fine-tuned Model" solution is: "Let's verify this step by step: 1. Starting with  $f(x) = x^2 + 3x$ , 2. Applying differentiation rules, 3. We get  $f'(x) = 2x + 3$ ".

Fig 9. The user interface allows us to visualize the full text output for human evaluation and comparison of the base to the fine-tuned model's output and run-time performance and log metrics.

# Conclusion

We have explored the significant challenges that large language models face in solving mathematical problems, from their struggles with symbolic manipulation to difficulties in logical reasoning. By implementing targeted training strategies, we aim to enhance their capabilities in this domain, opening new opportunities for application in technical fields and educational tools.

Future research would focus on enhancing the mathematical capabilities of large language models (LLMs) through two promising areas. First, algorithmic enhancements are crucial for integrating symbolic reasoning more effectively within LLM architectures, which involves refining their ability to understand and manipulate mathematical symbols and expressions. Second, the exploration of hybrid models, which combine LLMs with specialized mathematical solving systems, offers a significant opportunity. This approach leverages both the linguistic skills of LLMs and the precise numerical capabilities of math solvers to improve performance on complex tasks that require both textual understanding and numerical analysis.

# Bibliography

[1] Ahn, J., Verma, R., Lou, R., Liu, D., Zhang, R., & Yin, W. (Year). Large Language Models for Mathematical Reasoning: Progresses and Challenges. The Pennsylvania State University, Temple University.

[2] Xu, Y., Liu, X., Liu, X., Hou, Z., Li, Y., Zhang, X., Wang, Z., Zeng, A., Du, Z., Zhao, W., Tang, J., & Dong, Y. (Year). ChatGLM-Math: Improving Math Problem-Solving in Large Language Models with a Self-Critique Pipeline. Zhipu.AI, Tsinghua University.