Fine-Tuning Language Models for Math Problem Solving using LoRA

Popa Andrei Ionut
Universitatea Bucuresti
andreipopa842@yahoo.com

Toma Sabin Sebastian
Universitatea Bucuresti
tomasabin20@gmail.com

ABSTRACT

This paper explores the application of fine-tuning language models for solving mathematical problems from the GSM dataset using LoRA (Low-Rank Adaptation). We experiment with different models, including Facebook's OPT-125M, OPT-350M, and BERT-base, comparing their effectiveness in reasoning-based tasks. Our goal is to evaluate how LoRA improves performance while keeping computational costs low and analyze the limitations of smaller models in numerical reasoning tasks. Our findings show that while LoRA helps in fine-tuning efficiency, the models struggle to generate accurate solutions to mathematical problems.

1. INTRODUCTION

Recent advancements in large language models (LLMs) have demonstrated remarkable progress across various natural language processing (NLP) tasks, from text generation and translation to question-answering and code generation. Despite these breakthroughs, numerical reasoning and mathematical problem-solving remain significant challenges for even the most powerful models. Solving math problems, particularly those requiring multi-step reasoning, demands not only a strong understanding of arithmetic but also the ability to logically plan and execute sequential operations—capabilities that LLMs often struggle with.

In this study, we investigate the ability of transformer-based models to solve Grade School Math (GSM) problems, which serve as a benchmark for evaluating numerical reasoning and multi-step problem-solving. Rather than relying on massive-scale models such as GPT-4 or PaLM, we explore the effectiveness of fine-tuning smaller-scale models with parameter-efficient tuning techniques. Specifically, we leverage Low-Rank Adaptation (LoRA) to fine-tune three transformer models: Facebook's OPT-125M, OPT-350M, and BERT-base. Our goal is to assess how well these models can handle multi-step arithmetic reasoning when trained with limited computational resources.

LoRA has emerged as an effective method for reducing the computational and memory overhead associated with full fine-tuning. By injecting trainable low-rank updates into the frozen pre-trained weights of a model, LoRA enables efficient adaptation while preserving the benefits of pre-training. Given its efficiency, LoRA is a promising approach for adapting LLMs to domain-specific tasks, such as GSM problem-solving, without incurring the prohibitive costs of full fine-tuning.

Through our experiments, we analyze multiple factors, including:

- The relative performance of different model sizes when fine-tuned with LoRA.
- The ability of each model to handle multi-step mathematical reasoning.
- The trade-offs between model size, fine-tuning efficiency, and accuracy.

By comparing these models, we aim to provide insights into the feasibility of using smaller, fine-tuned transformers for math problem-solving, highlighting both the strengths and limitations of LoRA-based adaptation. Our findings contribute to the broader discussion on parameter-efficient fine-tuning and its applicability to reasoning-intensive NLP tasks.

2. DATASET

In our dataset, we have 2 columns: question(math word problems) and answer(step-by-step solutions). It includes basic arithmetic, proportion, division and multi-step problem solving, where the solutions are structured with intermediate calculations, often using inline explanations. The dataset is important for our study because it aligns well with Grade School Math problems, ideal for evaluating logical reasoning capabilities and provides a benchmark for measuring how well LoRa fine-tuning improves numerical problem solving.

Example data:

Question: Natalia sold clips to 48 of her friends in April, and then she sold half as many in May. How many clips did Natalia sell altogether in April and May?"

Answer:"Natalia sold 48/2 = ■48/2=24■ 24 clips in May. Natalia sold 48+24 = ■48+24=72■ 72" clips altogether in April and May.

This dataset allows us to test how well language models can handle logical reasoning and structured numeric computations.

3. MODELS AND FINE-TUNING APPROACH

3.1 OPT-125M

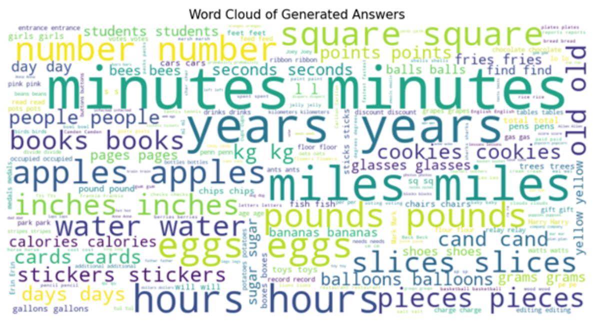
The OPT(Open Pre-trained Tranformer) family of models was developed by Meta AI as an open-source alternative to proprietary large language models. OPT-125M is the smallest variant, containing 125 million parameters, designed as a lightweight model for research and experimentation.

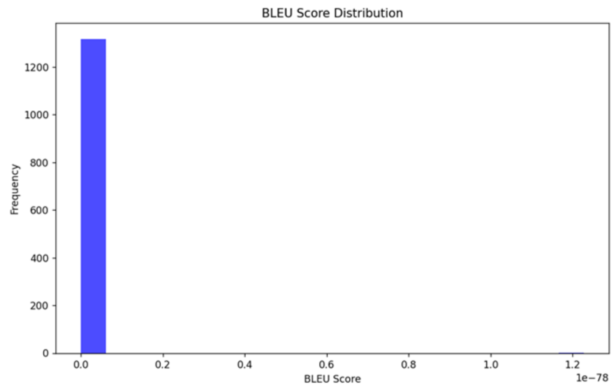
This model is based on the Transformer architecture, implements decoder-only architecture, similar to GPT-2/GPT-3, uses layer normalization, rotary positional embeddings(RoPe) and casual-attention. It is pre-trained using autoregressive language modeling(predicting the next token in a sequence).

It is pre-trained on a diverse dataset, including web text, books and other publicly available sources, trained with using FP16 precision to optimize efficiency and uses a GPT-like training objective: predicting the next token given the previous context.

Key features of my model:

- 125m - Lightweight transformer(125M params)
- Batch size = 2(small batch size, memory-efficient)
- Learning size = 5e-5(balanced for fine-tuning stability)
- Weight decay = 0.01(helps prevent overfitting)
- Epochs = 3(moderate training time)
- Sequence length = 512 tokens(ensures reasonable memory usage)
- BLEU score = 0.000 (used for evaluation, measures text generation quality)





Facebook/opt-125m and LoRa

LoRa (Low-Rank Adaptation) is a parameter-efficient fine-tuning (PEFT) method that allows large pre-trained models to be adapted to new tasks with minimal computational cost. LLMs have billions of parameters, making fine-tuning computationally expensive. LoRa reduces the number of trainable parameters by introducing low-rank matrices into transformer layers, keeping most of the model frozen, that results in lower memory usage and faster training. Combining LoRa with OPT-125M enables efficient fine-tuning without requiring extensive computational resources. LoRa makes it feasible to adapt OPT-125M for custom tasks like GSM reasoning.Great for resource-constrained environments where full fine-tuning is impractical. Future directions: Applying LoRA to larger OPT models (OPT-1.3B, OPT-6.7B) for more complex tasks.

Key features of my model:

- LoRa rank = 8(Controls memory-efficient)
- LoRa alpha = 16(Adjusts learning magnitude for LoRa layers)
- LoRa dropout = 0.1(Prevents overfitting in LoRa layers)
- Task type = TaskType.CasualLM
- For the facebook/opt-125 model hyperparameters, were used the same as in previous slide.
- BLEU score = 0.000

3.2 OPT-350M

OPT-350M is a larger variant with 350 million parameters, designed for improved efficiency in reasoning tasks. It is part of the Open Pretrained Transformer (OPT) series developed by Facebook AI, aimed at providing a lighter alternative to larger transformer models while maintaining strong language modeling capabilities. This model was pre-trained on a diverse dataset, enabling it to generalize well across different NLP tasks.

To fine-tune this model for math problem-solving, we applied proxy fine-tuning to adjust the model to this specific task without modifying all its parameters. Additionally, LoRA (Low-Rank Adaptation) was used to optimize memory efficiency by fine-tuning only a small subset of the model's layers (*q_proj, v_proj*) while keeping most of the model frozen. This approach allows for efficient adaptation without the need for high computational resources.

The training setup for fine-tuning OPT-350M on the GSM dataset was as follows:

- Loss function: Cross-entropy, applied at the token level to improve sequence-level predictions.
- Training epochs: 5, ensuring that the model had sufficient exposure to the dataset without overfitting.
- Batch size: 16, chosen to balance computational efficiency and training stability.
- Gradient accumulation: 4 steps, allowing for an effectively larger batch size while optimizing GPU memory usage.
- Optimizer: AdamW with a learning rate of 5e-5, ensuring gradual weight updates and preventing instability.
- Evaluation metric: Exact match score, which checks whether the model-generated response perfectly matches the correct solution.

Despite the advantages of LoRA in reducing computational costs, several challenges were encountered during fine-tuning. The model exhibited inconsistent numerical reasoning, often struggling with multi-step arithmetic operations and failing to break down problems into structured steps. Additionally, instead of developing reasoning capabilities, the model frequently memorized training examples, applying incorrect solutions to new problems. Another issue was the model's limited ability to generalize, as it struggled to adapt to variations in problem phrasing.

A particularly notable challenge was the tendency of OPT-350M to generate the same numerical answer across multiple questions. This suggests that the model was not fully engaging in the reasoning process but rather relying on learned patterns that did not generalize well to novel problems. These findings indicate that while LoRA provides an efficient method for fine-tuning, additional techniques such as chain-of-thought prompting, structured dataset design, or reinforcement learning may be required to improve mathematical reasoning performance.

3.3 BERT-Base

BERT-Base is a bidirectional transformer model pre-trained by Google, designed for deep text understanding. It consists of 12 layers, 768 hidden units, and 110 million parameters. Unlike traditional unidirectional language models, BERT processes text in both directions, considering both preceding and following words to better understand context. This makes it particularly useful for classification and question-answering tasks. BERT was pre-trained using two main objectives:

- **Masked Language Modeling (MLM)**: The model learns to predict missing words in a sentence by being trained on text where random words are masked. This helps BERT develop a deep contextual representation of language.
- **Next Sentence Prediction (NSP)**: The model learns to determine whether two given sentences are logically connected. This improves its ability to handle tasks requiring sentence relationships, such as question answering.

To adapt BERT for solving math problems, we reformulated problem-solving as a classification task. Instead of generating numerical answers directly, we assigned unique labels to each distinct answer in the dataset. This approach transforms the math problem-solving task into a multi-class classification problem, allowing the model to select from a predefined set of possible answers rather than computing them.

LoRA (Low-Rank Adaptation) was used to fine-tune BERT efficiently. LoRA updates only a subset of the model's parameters, significantly reducing the computational cost of fine-tuning while maintaining strong task-specific adaptation. By modifying only certain layers, LoRA makes it feasible to train large-scale transformer models on resource-limited hardware.

The training setup for fine-tuning BERT-Base on the GSM dataset was as follows:

- **Dataset**: GSM, split into training, validation, and test sets.
- **Loss function**: Cross-entropy, optimized for classification problems.
- **Training epochs**: 5, ensuring sufficient exposure to data without overfitting.
- **Batch size**: 16, selected to balance memory efficiency and model performance.
- **Optimizer**: AdamW with weight decay of 0.01, which applies regularization to prevent overfitting.
- **Evaluation metric**: Accuracy score, measuring the percentage of correctly classified answers.

Challenges and Observations: While BERT-Base demonstrated strong text comprehension capabilities, several issues arose when applying it to mathematical reasoning tasks:

- **Pattern recognition over reasoning**: Instead of logically solving problems, the model often relied on recognizing patterns from training data, leading to incorrect predictions when faced with slightly reworded questions.
- **Limited numerical understanding**: Since BERT was not explicitly trained on arithmetic tasks, it struggled to perform multi-step reasoning, often misclassifying problems that required deeper logical steps.

- **Copying input numbers**: In some cases, the model extracted a number from the input question and presented it as the answer, rather than deriving a correct solution.
- **Lack of interpretability**: Unlike step-by-step reasoning models, BERT provides no insight into how it arrives at a classification decision, making it difficult to debug incorrect predictions.

These findings highlight that while BERT can be adapted for structured classification problems, its ability to solve mathematical reasoning tasks remains limited. Future improvements could involve integrating hybrid models that combine deep learning with symbolic solvers, or using chain-of-thought prompting techniques to guide the model through intermediate reasoning steps.

4. RESULTS AND CHALLENGES

Our evaluation shows that no model generated a correct answer:

- Model 1: The outputs contain long, uniform, repetitions of words, stuck in repetitive loops of generic numbers, special characters and words. Suggests the issue where the model fails to produce coherent sentences.
- Model 2: It generated the same pattern of answers, loop of generic numbers and words, NOT special characters.

The model's structure remains limited, as both versions exhibit repetitive token behavior, indicating challenges in diversity of text generation. LoRa introduced some domain-specific influence since more words like "download/miles/boots" might relate to the fine-tuning dataset.

Example Question: Gretchen has 110 coins. There are 30 more gold coins than silver coins. How many gold coins does Gretchen have?

True answer: 70

Model 1 answer: 110110110110110110110110110

Model 2 answer: 3030303030303030303030303030

There were also cases where model generated words or special characters instead of numbers, like: day day day day day day day day day day day day day ; ; **;============= ; bees bees bees bees bees bees bees

Our experiments revealed several limitations in using LLMs for math problem-solving:

- OPT-350M consistently produced the same number for all questions, failing to generalize solutions.
- BERT-Base copied parts of the question and extracted random numbers instead of computing results.
- Smaller models struggle with multi-step reasoning, indicating the need for larger models with better fine-tuning strategies.
- Resource limitations restricted the use of larger, more capable models like OPT-1.3B or GPT-4.

These findings suggest that fine-tuning alone is insufficient for mathematical reasoning. More advanced techniques, such as chain-of-thought prompting and reinforcement learning, may be required for better performance.

Example

Question: Janet's ducks lay 16 eggs per day. She eats three for breakfast every morning and bakes muffins for her friends every day with four. She sells the remainder at the farmers' market daily for 2 dollars per fresh duck egg. How much in dollars does she make every day at the farmers' market? True Answer: 18 Predicted Answer:

Model 3: Q: Janet's ducks lay 16 eggs per day. She eats three for breakfast every morning and bakes muffins for her friends every day with four. She sells the remainder at the farmers' market daily for 2 dollars per fresh duck egg. How much in dollars does she make every day at the farmers' market? A:505050505050505050

Model 4: Q: Janet's ducks lay 16 eggs per day. She eats three for breakfast every morning and bakes muffins for her friends every day with four. She sells the remainder at the farmers' market daily for 2 dollars per fresh duck egg. How much in dollars does she make every day at the farmers' market? A: Janet's ducks lay 16 eggs per day

5. CONCLUSION AND FUTURE WORK

The models do not fully understand the step-by-step reasoning needed for solving math problems.

5.1 Computing Limitations

Larger models such as GPT-4 or LLaMA-13B would likely perform better but require high-end GPUs and large-scale computing resources. Training a bigger model from scratch is not feasible without specialized hardware, which we do not currently have. These challenges show that while our models can be improved, real progress requires more powerful resources and more advanced fine-tuning techniques.

5.2 What Could Improve Results?

- Using a larger and more powerful model, but these require much more computing power than we currently have.
- More advanced fine-tuning methods, like reinforcement learning or instruction tuning.
- Better dataset preprocessing to help the model learn the correct patterns.

Our project explored fine-tuning language models to solve math problems from the GSM dataset using LoRA for efficient adaptation. While our models did not perform as expected, this project highlighted important challenges in applying LLMs to structured math problems. Future improvements require more powerful models, smarter training techniques, and better data handling to truly enhance problem-solving capabilities.

6. REFERENCES

https://openreview.net/forum?id=dribhnhm1idiscussion - Tuning Language Models by Proxy

https://openreview.net/forum?id=dribhnhm1idiscussion - Supplementary work

https://huggingface.co/facebook/opt-350m

https://huggingface.co/facebook/opt-125m

https://huggingface.co/meta-llama/Llama-3.2-1B

https://github.com/huggingface/peft/issues/1589

https://huggingface.co/WizardLMTeam/WizardMath-7B-V1.1