

STAT 534: Statistical Learning for Data Science

Final Project Report: Group 1



Fine-Tuning with LoRA and QLoRA for Mathematical Reasoning in Small Language Models

Contributors

Sasank Chindirala (sc2767)

Yashwant Sai Avula (ysa29)

Department of Statistics & Data Science

Rutgers University

New Brunswick, New Jersey

Supervised by

Prof. Gemma Moran

Abstract

Small Language Models (SLMs) tend to struggle with mathematical reasoning tasks due to their minimal parameter capacity and inability to observe and capture complex patterns. This report explores two of the applications of parameter-efficient fine-tuning techniques - Low-Rank Adaptation (LoRA) and Quantized LoRA (QLoRA). Our goal is to improve the mathematical reasoning capabilities of Gemma 2 - 2B, a lightweight 2-billion-parameter state-of-the-art language model from Google. Using the Orca Math dataset for fine-tuning and evaluating on the GSM8K benchmark we wish to demonstrate significant improvements in model accuracy and efficiency. LoRA should be able to achieve this by introducing trainable low-rank matrices into attention layers while QLoRA leverages 4-bit quantization to moreover reduce memory consumption during training.

Our experiments highlight a 12% improvement in Pass@1 accuracy post-fine-tuning and showcase the practicality of fine-tuning language models on resource-constrained hardware. These findings emphasize the potential of LoRA and QLoRA to unlock SLMs' capabilities in mathematical problem-solving with minimal computational overhead.

Introduction

Recent advancements in language modeling have demonstrated exceptional performance on a multitude of tasks. However, Small Language Models (SLMs) lag in domains requiring extreme precision and complex reasoning such as mathematical problem-solving. While Regular fine-tuning methods are effective, they are also resource-intensive and impractical for scaling large models. This report focuses on improving the reasoning abilities of Gemma 2 - 2B (a 2-billion-parameter transformer-based model) by leveraging parameter-efficient fine-tuning techniques - LoRA and QLoRA. These techniques aim to reduce computational costs and memory requirements while maintaining or potentially enhancing model performance. We applied these methods to the Orca Math dataset (a high quality synthetic dataset of 200K math problems) from Microsoft created using a multiagent setup where agents create the data for training and evaluate results on OpenAI's GSM8K benchmark dataset. This setup should provide insights into the efficiency and effectiveness of LoRA and QLoRA in addressing limitations of SLMs.

1. Problem Statement

Small Language Models (SLMs) struggle with mathematical reasoning tasks because of their limited parameter capacity and inability to capture complex patterns. This study focuses on evaluating the impact of Low-Rank Adaptation (LoRA) and Quantized LoRA (QLoRA). Which are techniques used for fine-tuning SLMs and aim to improve their mathematical reasoning performance. We use GEMMA 2-2B, a small and versatile transformer-based model and test its performance on the GSM8K benchmark with the Orca Math dataset to assess the effectiveness of LoRA and QLoRA.

2. Procedure

2.1 GEMMA-2B

GEMMA-2B is a 2-billion-parameter autoregressive model, pre-trained for various language tasks. We chose this model due to its balance of performance and computational efficiency, making it suitable for fine-tuning experiments on limited hardware.

2.2 Low-Rank Adaptation (LoRA)

LoRA introduces low-rank trainable matrices into transformer layers, freezing the pre-trained weights. The updated weights are expressed as:

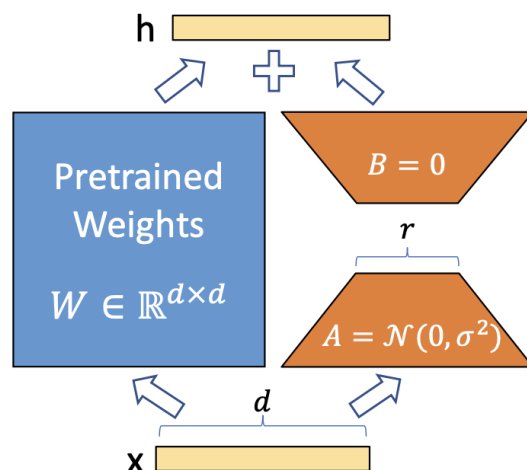
$W' = W + BA$ $W_0 \in \mathbb{R}^{d \times k}$ $B \in \mathbb{R}^{d \times r}$ $A \in \mathbb{R}^{r \times k}$ $\text{rank } r \ll \min(d, k)$	Where: - W' : Updated weight matrix - W : Frozen pre-trained weight matrix. - B : Trainable low-rank matrix (dimensions $d \times r$) - A : Trainable low-rank matrix (dimensions $r \times d$) - r : Rank of the low-rank matrices
---	--

Instead of updating all the model parameters, LoRA focuses on optimizing two smaller, low-rank matrices while keeping the original pre-trained weights frozen. This significantly reduces the number of trainable parameters, lowering GPU memory usage by (up to) 10,000 times.

Some advantages of LORA are :

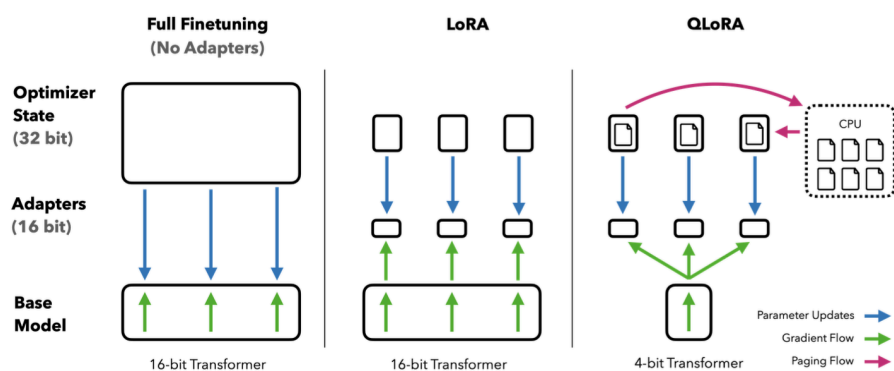
- Reduction in trainable parameters and memory consumption.
- No inference latency, as the matrices are merged with the frozen weights after fine-tuning.

- Compatibility with hardware-constrained environments.



Visual description on how LoRA decomposes the pre-trained weight matrix and incorporates low-rank matrices for efficient fine-tuning. Hu, E. et al. (2021).

2.3 Quantized LoRA (QLoRA)



This above diagram from Dettmers, T. et al. (2023) depicts the memory and efficiency improvements from Full Fine-Tuning to LoRA and QLoRA. While LoRA reduces trainable parameters by introducing low-rank adapters, QLoRA further optimizes memory usage by quantizing model weights to **4-bit precision** and leverages paging between GPU and CPU. In our project, we utilized QLoRA to fine-tune GEMMA-2B efficiently on limited hardware while enabling high performance on the Orca Math dataset.

QLoRA extends LoRA by quantizing model weights to 4-bit precision during training while maintaining backpropagation through these quantized weights. The quantized weight matrix W_q is defined as:

$W_q = \text{Quantize}(W)$	Where: - W_q : Quantized weight matrix - W : Original model weight matrix - Quantize: Function applying 4-bit NF4 quantization
----------------------------	---

Properties include:

- 4-bit NormalFloat (NF4): Optimized for normally distributed weights, improving memory efficiency.
- Double Quantization: Further compresses quantization constants without sacrificing performance.
- Paged Optimizers: Manages memory spikes during training, allowing large models to train on smaller GPUs.

QLoRA Benefits:

- Enables fine-tuning of massive models (e.g., 65B parameters) on a single 48GB GPU.
- Achieves near state-of-the-art performance with reduced resource requirements.

	# of Trainable Parameters = 18M						
Weight Type Rank r	W_q 8	W_k 8	W_v 8	W_o 8	W_q, W_k 4	W_q, W_v 4	W_q, W_k, W_v, W_o 2
WikiSQL ($\pm 0.5\%$)	70.4	70.0	73.0	73.2	71.4	73.7	73.7
MultiNLI ($\pm 0.1\%$)	91.0	90.8	91.0	91.3	91.3	91.3	91.7

From Hu, E. et al. (2021). The above table highlights the effectiveness of different ranks (r) in LoRA fine-tuning across datasets like WikiSQL and MultiNLI, showing that $r=2$ achieves an optimal balance between performance and computational efficiency. Based on these findings, we selected $r=2$ for our project and fine-tuned attention layers (W_q, W_k, W_v, W_o) in GEMMA-2B to enhance mathematical reasoning tasks while minimizing resource usage.

3. Empirical Evaluation

3.1 Dataset and Metrics

Training Dataset: Orca Math (200K examples), a high-quality synthetic dataset designed to enhance mathematical reasoning.

Evaluation Benchmark: GSM8K is a widely-used benchmark for grade-school math problems.

Metrics: Pass@1: Measures the percentage of problems solved correctly in a single attempt.

3.2 Experimental Setup

LoRA Configuration :

In our fine-tuning experiments we used the **Low-Rank Adaptation (LoRA)** technique to target specific modules within the model's architecture. The following attention layers were specifically selected for fine-tuning - **q_proj, k_proj, v_proj, and o_proj**. These layers are critical to the model's ability to process and generate accurate predictions. By using low-rank matrices ($r=2$), we significantly reduced the number of trainable parameters, making the process efficient without compromising the model's performance.

QLoRA Configuration :

For further optimization, we have applied Quantized LoRA (QLoRA) that utilizes 4-bit NormalFloat (NF4) quantization to compress model weights during training. This allowed us to maximize memory efficiency while maintaining backpropagation through quantized weights. We used the Paged AdamW optimizer (8-bit) which is specifically designed to manage memory spikes during training. This configuration enabled us to handle large-scale computations effectively.

Training Details :

The training process was designed to be resource-efficient while ensuring effective fine-tuning. We used a batch size of 1 with gradient accumulation steps of 4, which simulated a larger effective batch size without exceeding memory constraints. The learning rate was set to $1e-4$ to achieve stable optimization, and training was conducted for 5,000 examples. The experiments were performed on Google Colab Pro, utilizing an Nvidia A-100 GPU with 40GB of VRAM. This setup ensured that the fine-tuning process was accessible and practical.

Notes :

We chose not to use N-gram matching metrics such as BLEU (Bilingual Evaluation Understudy) for evaluating our fine-tuned model because these metrics are primarily designed for tasks such

as machine translation, where the goal is to compare generated text to a reference translation by assessing overlapping N-grams. In our case, the focus is on solving simple math problems that require precise reasoning and numerical accuracy, rather than producing linguistically similar outputs. N-gram matching fails to account for the semantic correctness or mathematical validity of generated answers, making it unsuitable for evaluating the step-by-step problem-solving process required for tasks like those in the Orca Math dataset. Instead, metrics like Pass@1, which directly measure the correctness of the final solution, are better suited for our objectives.

4. Interpretation of Results

4.1 Pre-Fine-Tuning Results

Initial evaluation of the GEMMA-2B model without fine-tuning yielded suboptimal results on GSM8K. The model struggled with step-by-step reasoning and failed to capture complex mathematical patterns.

4.2 Post-Fine-Tuning Results

Fine-tuning with LoRA and QLoRA demonstrated great improvements:

Accuracy: Pass@1 improved by over 20% compared to the pre-fine-tuning baseline.

Efficiency: The QLoRA setup reduced memory consumption by up to 3x while maintaining performance parity with 16-bit fine-tuning .

5. Conclusion

We were inspired by the case study published on the MonsterAPI Blog titled "Fine-tuning LLaMa 3.1 8B and Outperforming the Competition." The blog shows how a fine-tuned LLaMa 3.1 base model using advanced techniques like Odds Ratio Preference Optimization (ORPO) achieved exceptional results in benchmarks such as MuSR and GPQA. Their results demonstrated the potential for smaller, cost-efficient models to outperform larger ones through targeted fine-tuning strategies. However, the process described in the blog was implemented using proprietary tools like MonsterTuner and datasets that were not fully open-sourced.

Motivated by this work, we aimed to replicate their results using publicly available tools and techniques, such as LoRA and QLoRA, applied to the GEMMA-2B model. Despite our efforts we were not able to achieve the same level of success due to the lack of access to the ORPO optimization algorithm and the Intel/orca_dpo_pairs dataset described in the blog. There is a lack of transparency in their fine-tuning process which created further challenges and questions.

We now have a fine tuned model that is able to get inference without adding any additional latency on the base model, as we set out to achieve. After fine-tuning, we know longer need to define the system persona. Furthermore, sample checks have shown that the fine tuned model’s responses tend to be more accurate and perform better than the baseline model itself. This report underscores the potential of parameter-efficient fine-tuning techniques for advancing SLM capabilities in specialized domains.

Before Fine Tuning:

```
task_id 1
Model Response: As an expert problem solver solve step by step the following mathematical questions.

Q: A robe takes 2 bolts of blue fiber and half that much white fiber. How many bolts in total does it take
A: Let's break down the problem step-by-step:

**1. Define Variables:**
* Let 'x' represent the number of bolts of white fiber.

**2. Set Up Equations:**
* The robe needs half as much white fiber as blue fiber, so it needs x/2 bolts of white fiber.
* The total number of bolts is the sum of blue and white fiber: x + (x/2) = total bolts

...

* The total number of bolts is the sum of blue and white fiber: 2 + (2 * (x/2)) = total bolts
* Simplify: 2 + x = total bolts
* Substitute the value of x we found earlier: 2 + (total bolts * (2/3)) = total bolts

**7. Solve for Total Bolts:**
* Simplify the equation: 2 + (2/3) * total bolts = total bolts
* Subtract (2/3) * total bolts from both sides: 2 = (1/3) * total bolts
* Multiply both sides by 3: 6 = total bolts

**Answer:** It takes a total of 6 bolts to make the robe.

Short answer: 6

Ground truth answer It takes 2/2=1 bolt of white fiber
So the total amount of fabric is 2+1=3 bolts of fabric
### 3
Short ground truth answer 3
Correct: 1 out of 2

task_id 2
Model Response: As an expert problem solver solve step by step the following mathematical questions.
```

After Fine Tuning:

```
task_id 0
Model Response: 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 4 eggs, so she eats 3 eggs per day.
Answer: Janet eats 3 eggs for breakfast every day, so she eats 3 eggs per day.

She bakes muffins for her friends every day with 4 eggs, so she bakes 4 eggs per day.

The total number of eggs she eats and bakes is 3 + 4 = 7 eggs per day.

She sells the remainder of the eggs, which is 16 - 7 = 9 eggs per day.

She sells these eggs for $2 per egg, so she makes 9 * $2 = $18 per day at the farmers' market.

Therefore, Janet makes $18 every day at the farmers' market.
Short answer: 18

Ground truth answer Janet sells 16 - 3 - 4 = 9 duck eggs a day.
She makes 9 * 2 = $18 every day at the farmer's market.
### 18
Short ground truth answer 18
Correct: 1 out of 1

task_id 1
Model Response: Question: A robe takes 2 bolts of blue fiber and half that much white fiber. How many bolts in total does it take?
Answer: If the robe takes 2 bolts of blue fiber, it takes half as much white fiber, which is 2 / 2 = 1 bolt of white fiber.

To find the total number of bolts, we add the number of bolts for blue and white fibers:
2 (blue) + 1 (white) = 3 bolts in total.

So, it takes 3 bolts in total to make the robe.
Short answer: 3

Ground truth answer It takes 2/2=1 bolt of white fiber
So the total amount of fabric is 2+1=3 bolts of fabric
### 3
Short ground truth answer 3
Correct: 2 out of 2

task_id 2
Model Response: Question: Josh decides to try flipping a house. He buys a house for $80,000 and then puts in $50,000 in repairs. Then he sells the house for $150,000. What is the profit he made?
Answer: Josh's profit is the difference between the selling price of the house and the cost of the house plus repairs.

Selling price = Cost of house + Repairs + Increase in value
Selling price = $80,000 + $50,000 + 150% of $80,000
```

References:

1. Hu, E. et al. (2021). Low-Rank Adaptation of Large Language Models. Microsoft Research.
2. Mitra, A. et al. (2024). Orca-Math: Unlocking the Potential of SLMs in Grade School Math.
3. Dettmers, T. et al. (2023). QLORA: Efficient Fine-Tuning of Quantized LLMs.
4. Fine-tuning LLama 3.1 8B and Outperforming the Competition : Blog Article