Optimizing Smaller Language Models for Scientific Question Answering Using Parameter-Efficient Fine-Tuning

Anay Pardasani

Received April 01, 2025
Accepted June 29, 2025
Electronic access July 15, 2025

Although Natural Langfuage Processing (NLP) keeps breaking the boundaries of comprehending language, Large Language Model (LLM) computational costs remain a huge entry barrier. This project targets smaller language models (SLMs), for which we employed a use case of scientific question answering in physics, chemistry, and biology. We created a dataset of 13,768 four-option multiple-choice questions along with an explanation summing up to approximately 1.5 million tokens after pre-processing. With this data, we fine-tuned Phi 1.5 model (1.3 billion parameters) with Parameter-Efficient Fine-Tuning (PEFT) techniques, i.e., Low-Rank Adaptation (LoRA) and its quantized variant, QLoRA. Both updated the majority of the parameters and left a very small residue, which allowed for efficient adaptation on consumer-grade hardware like an NVIDIA RTX 3080 GPU. Fine-tuned model performed better than baseline Phi 1.5 on key metrics: BLEU score from 0.0027 to 0.132, ROUGE-L from 0.152 to 0.235, and BERT F1 from 0.5266 to 0.6002, driven by a boost in recall from 0.4711 to 0.6553 at the cost of a drop in average precision (0.61 to 0.57). Such gains validate PEFT to be able to enhance SLM's performance without the need for heavy computing resources, in favor of advanced NLP into environments like high school research. This paper depicts how PEFT-optimized SLMs can effectively meet domain-specific issues and offer a scalable, cost-efficient substitute for LLMs.

Introduction

Rapid growth of the Large Language Model (LLM) has ushered in a new paradigm for Natural Language Processing (NLP), one of unprecedented performance across a wide range of tasks like language generation, translation, question answering, and reasoning. The LLM scale law whereby model quality increases with size has fueled the spread of progressively larger models, to hundreds of billions of parameters. These massive models have established cutting-edge performance across a wide range of benchmarks, pushing the boundaries of what is possible in NLP. But the vast computational resources required to train and run these giants are a huge entry point for the majority of researchers and developers, limiting access to these powerful tools and hindering broader exploration of their possibilities. This has generated growing interest in small language models (SLMs) offering a lighter and resource-efficient but still significance-capability alternative. This piece is interested in enhancing SLM performance, or rather, in the scientific question answering domain, through targeted fine-tuning [1] from a high-quality dataset of chemistry, physics, and biology queries. Their success is owed to a variety of factors such as the transformer model, having large datasets at hand, and advances in training methodologies.

Fine-tuning, or specializing a pre-trained model to perform on a specific task or dataset, has become a de facto approach to achieve optimum performance on a specific domain. Fine-tuning large LLMs, however, is computationally costly, requiring huge amounts of computing resources that typically exceed existing hardware capabilities. This makes the need for efficient fine-tuning techniques along with investigating smaller, manageable models imperative. Techniques like Low-Rank Adaptation (LoRA) [2] and its quantized variant QLoRA [3] have been computation-light alternatives to fine-tuning larger language models. LoRA compresses trainable parameters by adding low-rank updates to the model weight matrices, in effect, compressing the model representation during fine-tuning. QLoRA takes efficiency further with quantization of the model weights, reducing the memory requirement and fine-tuning on consumer-grade hardware. These techniques have democratized LLM fine-tuning and allowed researchers and developers with limited resources to adapt these powerful models to their tasks. While parameter-reducing fine-tuning methods like QLoRA address the computational problem, consistency and reproducibility of results remain a prime concern.

Replicability of fine-tuning experiments is key to ensuring validity and replicability of results. Random seeding of initialization, data shuffling, and varying hardware used can influence the process of fine-tuning and, consequently, result in varying performance while executing differently. Further studies in the causes of issues with repeatability and creating means to cope with them are necessary to develop robust and stable fine-tuning pipelines. In this project, I have optimized SLM fine-tuning on Scientific Question Answering using Parameter-Efficient Fine-

Tuning (PEFT) which is a resource efficient technique. Also demonstrating that this technique improves the performance in SLMs.

Literature Review

The evolution of fine-tuning techniques for large language models (LLMs) has been driven by the need to adapt pre-trained models to specific tasks while mitigating the computational and environmental costs associated with traditional methods. Early fine-tuning approaches, as validated by the paper[4] , involved updating all parameters of a pre-trained model, such as BERT, to achieve high performance on downstream tasks. However, this method showed serious drawbacks when models grew to hundreds of millions or billions of parameters. In order to fine-tune a model with 1.5 billion parameters like GPT-2, for example, significant computing resources were needed, such as 24GB of VRAM for 32-bit precision, which frequently exceeded the capabilities of consumer-grade GPUs. This paper[5] refers to the problem of catastrophic forgetting was also brought to light, whereby complete fine-tuning deteriorated performance on irrelevant tasks since it overwrote general knowledge stored in the pre-trained weights. These difficulties highlighted the need for parameter-efficient fine-tuning (PEFT) techniques that may reduce resource requirements and preserve general knowledge while adapting large models to new tasks.

Initial efforts in PEFT focused on modifying specific subsets of model components to minimize the number of trainable parameters. One prominent approach[6] , involved the use of adapter layerssmall bottleneck layers inserted within transformer blocks. These layers typically required training only 3-4 % of the models parameters, making them significantly more efficient than full fine-tuning. Adapters proved effective in tasks such as cross-lingual transfer, where they enabled models to adapt to new languages without extensive retraining. However, two critical drawbacks[7] : sequential computation of adapter layers increased inference latency by 23-40%, and concurrent adaptation to multiple tasks led to a 14% performance degradation due to interference in shared adapter parameters. These limitations prompted further exploration into alternative PEFT strategies that could balance efficiency and performance.

Another early PEFT method, prefix tuning, was proposed[8]. This approach involved prepending trainable vectors, or prefixes, to input sequences, allowing the model to adapt to new tasks by optimizing only a small fraction (0.1-0.3%) of parameters. Prefix tuning achieved impressive results, retaining approximately 87% of full fine-tuning performance on models like GPT-2. However, prefix tunings effectiveness was highly sensitive to the length of the prefix vectors, with optimal performance often requiring 20-50 vectors per task. This dependency complicated deployment in multi-task scenarios, as managing variable prefix lengths across tasks introduced additional complexity and potential instability. Despite its efficiency, prefix tunings sensitivity to hyperparameter choices limited its scalability in diverse applications.

A simpler approach, BitFit, was introduced[9]. BitFit: Simple parameter-efficient fine-tuning for transformer-based masked language models. In Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers) (pp. 1-9), which focused exclusively on updating the bias terms of a model, constituting just 0.1-1% of total parameters. BitFit demonstrated competitive performance on classification tasks, where fine-tuning bias terms alone was sufficient to achieve reasonable accuracy. However, its limitations became evident in generative tasks, such as machine translation, where BitFit achieved only 61% of full fine-tuning BLEU scores. This performance gap highlighted the methods inability to capture the complex weight interactions required for tasks involving sequence generation, restricting its applicability to a narrow range of use cases.

The introduction of Low-Rank Adaptation (LoRA)[10] marked a significant advancement in PEFT. LoRA leverages low-rank matrix decomposition to freeze the pre-trained weight matrix and learn incremental updates where both A & B matrices have very small rank compared to the actual dimensions of the model. This approach typically involves training only 0.1-0.5% of parameters, offering substantial efficiency gains. Unlike adapters, LoRA introduces no additional inference latency, as the update is computed using existing matrix multiplication units. Furthermore, LoRAs use of independent matrices for each task ensures robust task isolation, preventing interference in multi-task settings. In the above paper it is demonstrated that LoRA achieved 87.2% of full fine-tuning accuracy on the GLUE benchmark, outperforming adapters (85.4%) and prefix tuning (85.2%). LoRA reduces the VRAM usage consumption upto times of the actual model making a highly practical solution for resource-constrained environments.

QLoRA[11] represents a sophisticated integration of LoRA's parameter-efficiency with low-bit quantization, specifically utilizing 4-bit NormalFloat (NF4). This synergistic approach is meticulously designed to drastically reduce the memory footprint and computational costs associated with fine-tuning LLMs. The core idea involves quantizing the vast majority of the frozen base model weights to a lower precision, typically 4-bit, while the small LoRA adapters are trained in higher precision. This strategy aims to maintain high task performance despite the aggressive quantization of the base model. QLoRA has demonstrated a revolutionary reduction in GPU memory requirements, representing a significant paradigm shift in LLM deployment. It enables the fine-tuning of a 65-billion parameter model on a single 48GB GPU. This is a staggering reduction from the over 780GB typically required for traditional 16-bit full fine-tuning, representing a memory reduction of over 94%. Importantly, when compared to a 16-bit fully fine-tuned baseline, this signifi-

cant memory reduction is accomplished without compromising runtime or predictive performance. This speeds up research cycles and allows for real-world applications in settings that were previously thought to be infeasible due to prohibitive resource limits by making it economically and practically possible to fine-tune and deploy powerful LLMs on commodity hardware.

QLoRA has been effectively applied to fine-tune Financial Large Language Models (FinLLMs), demonstrating its adaptability and robustness in a highly specialized and data-sensitive domain. Models fine-tuned with QLoRA exhibit a significant average increase in accuracy compared to baseline models across diverse financial datasets. This substantial quantitative improvement strongly validates QLoRA's effectiveness in addressing the unique challenges and complex data characteristics inherent to FinLLMs [12].

Methods

Dataset Preparation

For this project, the data used for it comprise 13768 chemistry, physics and biology questions. I have curated this dataset from online resource HuggingFace. It contains 6 columns namely for question, correct_answer, support and three distractors. The questions were first in Multiple choice question answer form. That is, there were four options for every question of which only one was correct. Every answer had a reason why it is the correct answer. After performing the data cleaning and preprocessing steps, I had bunched the correct answer and reasoning in a single column as "ground truth". Now the questions were not labeled. Named Entity Recognition was performed on each data point to categorize them into Physics, Chemistry and Biology. This new column and question column was used for the dataset during Fine-tuning.

Supervised Fine-Tuning

Phi 1.5 [13] has about 1.3 billion parameters. Gradients are back-propagated through the entire model and all the weights are adjusted in consequence. Attention2 mechanism helps the model understand the contextual meaning among words. Flash Attention was also used within this model to improve training speed. It reduces memory overhead and avoids computation of large matrices using block sparse computation techniques. Phi-.5 achieves performance comparable to models that are five times larger, such as Llama2-7B, Falcon-7B, and Vicuna-13B, across nearly all benchmarks. In some cases, it even surpasses most non-frontier LLMs on more complex reasoning tasks. I have applied our training data with readability and consistency prompts to our task. During the period of time when the experiment was conducted, there werent as many SLMs as there are now. Also significant innovation lies in its training data. Phi-1.5 was

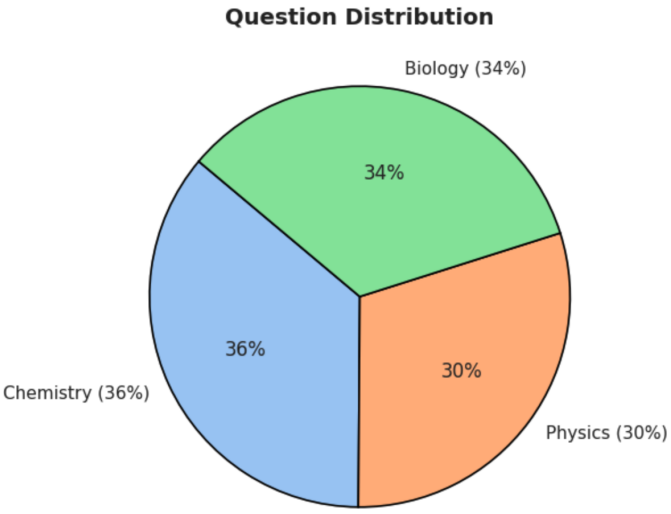

Fig. 1 Subject Wise distribution of question in the dataset

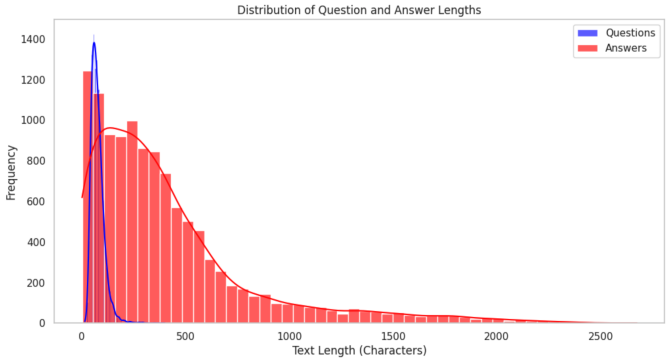

Fig. 2 Distribution of Question and Answer Lengths in a Q&A Dataset

trained on 150 billion tokens, with a crucial 80% derived from a newly created synthetic, "textbook-like" dataset, comprising approximately 20 billion tokens. This dataset was meticulously curated to teach common sense reasoning and general world knowledge, covering topics such as science, daily activities, and theory of mind. This contrasts sharply with models trained on vast, uncurated internet-scale data, which can contain noise, incomplete information, and biases.Because of the training process and the good performance of Ph1.5 against LLMs, I have chosen this model. Later to compare the performance of Phi1.5 I have also fined Tinyllama 1.1B and Pythia 1.4B.

Pass all the data points through a function and convert to a Question and Answer prompt data point. Divide the dataset into a train and test split of the ratio 8:2. The dataset has approximately 1.5 million tokens. After converting the dataset points into question and answer prompt structure, I tokenized it based on the models tokenizer. Fine-tuning the whole model will be tedious and time-consuming. Also, according to the size of the dataset, it can cause overfitting. Also taking into account the capacity of the hardware, I had employed Parameter

Efficient Fine-tuning (PEFT), which is a form of instruction fine-tuning [14]. It leverages vast processing in training a language model, particularly for the full LLM fine-tuning.

Naive hardware cannot cope with it because memory allocation would not only be required for the model but also for main parameters during training. PEFT overcomes this by having the capacity to successfully "freeze" the other parameters and adjust only a fraction of them. This minimizes the number of trainable parameters, thus decreasing the memory cost and avoiding catastrophic forgetting. PEFT does not alter the original LLM weights, losing no knowledge gained, unlike full fine-tuning. Fine-tuning for various tasks, this strategy is effective to solve storage issues. There are a few effective methods of parameter fine-tuning. Low-Rank Adaptation LoRA and QLoRA are the most widely used ones.

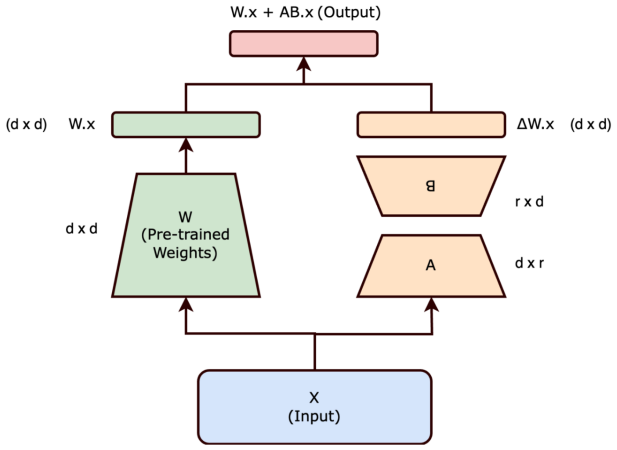

Fig. 3 Process of adding new Parameters to the SLM

LoRA

This method is based on the assumption that task-specific adaptation changes of model weights are also of low intrinsic rank. Rank is defined as the maximum number of linearly independent columns or rows present in a matrix. LoRA ((E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, L. Wang, W. Chen, LoRA: Low-rank adaptation of large language models. Int. Conf. Learn. Represent. 1, 3 (2022).)) exploits this fact and learns low-rank decomposition matrices to approximate updates of dense layer weights, but the initial pre-trained weights remain unchanged. This manner, LoRA introduces a lightweight and efficient solution for pre-trained model adaptation without the need to change or retrain the entire parameter set, as depicted in Figure 3

We hypothesize that the updates on weight at adaptation time have low "intrinsic rank." For a pre-trained weight matrix: W0 Rdk, update is limited by writing the modification as a low-rank factorization: $W_0 + \Delta W = W_0 + BA$, where: $B\varepsilon R^{dr}, AR^{rk}, rankr << min(d,k)$. Here d represents input dimensionality and k represents output dimensionality of the model. W_0 is held constant and never updated during training, and A and B are learnable parameters. Both W_0 and the update $\Delta W = BA$ are multiplied by the same input, and the outputs are added element-wise. With the computation: h = W0x, the adjusted forward pass is: $h = W_0x + \Delta W_x = W_0x + BA_x$.

QLoRA

In inference to weighted adaptations, adapter matrix was supplied. I have also utilized QLoRA [3]. This becomes memory-efficient on LoRA with quantization addition to LoRA adapters. This reduces precision on adapter weights to a lesser bit size, e.g., 4-bit from 8-bit, reducing memory usage and storage space dramatically. Compared to LoRA quantizing weights 8-bits, QLoRA saves the pre-trained model in GPU memory with quantized 4-bit weights. Despite this loss of accuracy, the performance of QLoRA is on par with that of LoRA. The fine-tuning metrics of the model are provided below.

Table 1 Details of Fine-tuning Parameters

Parameters	Values
LoRA Rank	16
LoRA Alpha	16
LoRA Dropout	0.05
Learning Rate	2.00E-04
Batch Size	16

LoRA Rank (16) is the rank of the low-rank matrices,It's a trade-off between efficiency and information retention. LoRA Alpha (also 16) is a scaling factor to control the effect of the low-rank updates on the model weights. LoRA Dropout (0.05) is used for regularization and randomly drops 5% of the connections at training time to prevent overfitting. Learning Rate (2e-4) controls the speed of parameter update, stabilizes training, and prevents overshooting. Lastly, Batch Size (16) specifies how many samples are trained in parallel at one time, sacrificing memory for model convergence.

Training Process

The dataset consisted of 13768 samples, out of which 11448 samples were used for Fine-tuning the models. 1160 samples were used for validation and the rest 1160 were used for testing. For future purposes, we can look for another dataset to evaluate the model's performance. Using the below steps I have loaded the model. The QLoRA methodology, as described by and adopted in this study, involves two key steps:

- Quantization of the Base Model: The pre-trained model (e.g., Phi 1.5) is loaded with its weights quantized to a

lower precision, specifically 4-bit NormalFloat in this case. This step significantly reduces the memory footprint of the base model, which is the ”Q” (Quantized) aspect of QLoRA.

- Application of LoRA Adapters: Low-Rank Adaptation (LoRA) adapters are then introduced to this 4-bit quantized base model. These LoRA adapters are the only parameters that are updated during the fine-tuning process. The hyperparameters detailed in Table 4 of the manuscript (LORA Rank, LORA Alpha, LORA Dropout) pertain to these LoRA adapters operating within the QLoRA framework.

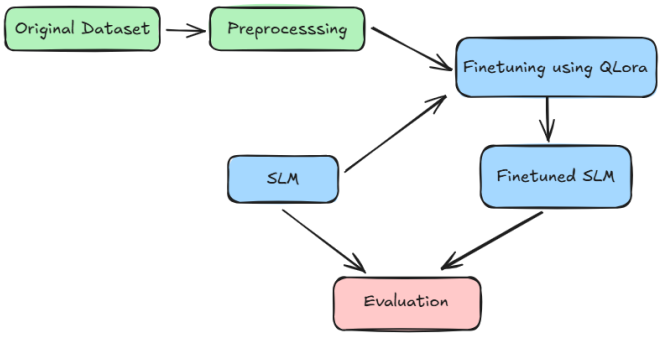

Fig. 4 Methodology Diagram

After loading, using HuggingFaces SFT trainer library I have trained the models for 2 epochs.Total Training samples were 11448. My batch size was 5. So approximately 2290 steps per epoch. The term steps refers to the number of parameter update iterations performed during training.

The training loss decreases consistently, and validation loss stabilizes after the first 1000-1500 steps.The Phi-1.5 model shows good convergence. The gap between training and validation loss is small, indicating minimal overfitting and good generalization.

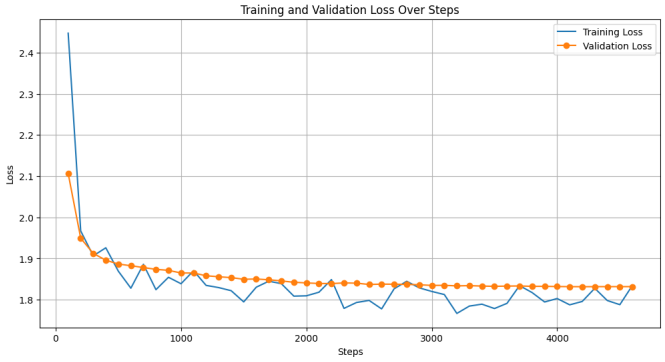

Fig. 5 Phi 1.5 Learning curves for training and validation loss over epochs

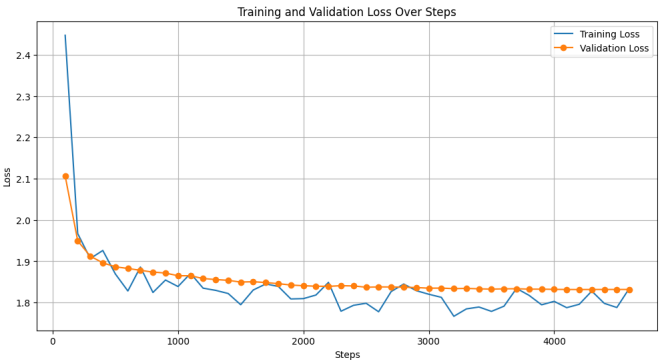

Fig. 6 Pythia1.4B Learning curves for training and validation loss over epochs

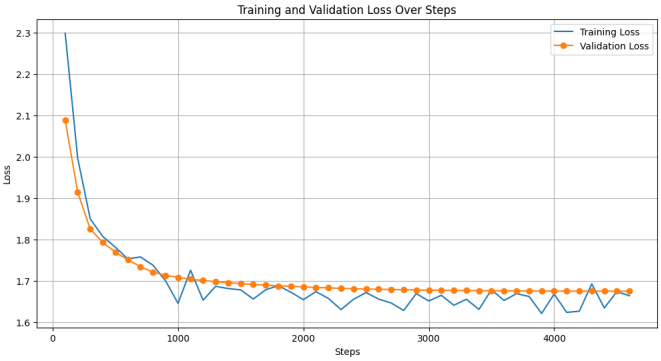

Fig. 7 TinyLlama1.1B Learning curves for training and validation loss over epochs

The loss curves are relatively smooth. Validation loss stays close to the training loss with a gradual downward trend. . Slightly higher validation loss at later steps suggests that further fine-tuning or regularization might improve performance.

TinyLlama achieves the lowest training and validation losses among the three models. Both curves steadily decrease and remain close.TinyLlama achieves the lowest training and validation losses among the three models. Both curves steadily decrease and remain close. The following measures were taken for avoiding overfitting:

- **LoRA Dropout:** A LoRA-specific dropout rate of 0.05 was applied during the training of the adapters, as detailed in Table 4 of the manuscript. Dropout is a widely used regularization technique that randomly sets a fraction of neuron activations to zero during training. This prevents neurons from co-adapting too much and forces the network to learn more robust and generalizable features.

- **Limited Training Epochs / Early Stopping:** For the Phi 1.5 model, training was conducted for only two epochs.1 This limited exposure to the training data can act as an implicit form of early stopping, preventing the model from over-training. The learning curves for all models show

the validation loss plateauing, suggesting that continued training beyond this point might not yield substantial improvements in generalization and could increase the risk of overfitting. If explicit early stopping criteria based on validation performance were employed for Pythia 1.4B and TinyLlama 1.1B, this would further contribute to overfitting prevention.

- **Dataset Characteristics:** The dataset, comprising 13,768 question-explanation pairs (approximately 1.5 million tokens) , provides a substantial amount of data for fine-tuning SLMs using PEFT. While not as massive as pre-training corpora, its size is adequate for effective adaptation without excessive risk of memorization when combined with PEFT.

The combination of these strategies reducing trainable parameters via PEFT, applying dropout, and limiting training duration (implicitly or explicitly) contributed to the observed stable validation performance and mitigated overfitting.

Results

After tuning the model we got a training loss of 1.5 after 2 epochs. Previously we had split 20% of the data for testing. We have the ground truth with us. We added two more columns, one contained results from original phi 1.5, and the other contained results from our tuned model. For testing we are using BLEU, ROGUE and BERT scores. Let's understand these metrics in detail.

BLEU

The BLEU[15] score is one of the most frequent measures for evaluating the quality of machine translated output in relation to a reference text. It calculates the n-gram precision (unigram, bigram, trigram, etc.) of the output generated by the machine relative to the reference. It also imposes a penalty for brevity so that very short outputs are not penalized. Higher BLEU scores indicate greater similarity between generated text and ground truth. The score of the fine-tuned model was 0.132 and that of the Parent model was 0.0027. However, BLEU only considers exact word-to-word similarity and not synonyms or semantic matches, so its application may be limited for the evaluation required in tasks demanding a finer type of judgment.

ROUGE

ROUGE is broadly used to score text summarization using the assessment of n-gram overlap, word sequence overlap, or longest common subsequences between the summary and reference. There are versions like ROUGE-N that focus more on n-gram overlap (e.g., ROUGE-1 for unigrams, ROUGE-2 for bigrams), while ROUGE-L uses the longest common subsequences. This measure values recall and, therefore, finds its usefulness especially in uses where the recovery of all of the information related to the purpose is critical. By examination of various aspects of text overlap, ROUGE provides a comprehensive estimate of the generated text's coverage of the reference. The table below shows that the fine-tuned model outperforms the real model across all variants.

BERT

Accuracy, in its conventional sense, measures the proportion of correct predictions out of the total predictions. When it comes to classification tasks (like spam mail identification) or short answer for questions, Accuracy is a good metric. Nonetheless, the goal of text production tasks like translation, summarization, and conversational answers is to generate text that is logical, meaningful, and semantically significant. There is typically more than one distinct "correct" outcome in these situations.. Many different phrases or sentences can convey the same meaning, and a word-for-word match with a reference text is rarely the sole indicator of quality, nor is it always expected or desirable. BERT score uses the context embeddings from transformer models like BERT to calculate the semantic similarity between reference and generated text. BERTScore differs from BLEU and ROUGE in the sense that it doesn't search for exact matches but calculates how well the generated text aligns with the reference based on meaning by comparing token embeddings. It provides precision, recall, and F1 scores, which enables a thorough exploration of the semantic fidelity of the model. BERTScore is especially useful in evaluating scientific or technical material, where the output sentence may be varied word for word but similar in overall meaning. For BERT score, first we have to find the token embeddings and then similarity calculation. Token Embedding: Both the generated and reference texts are tokenized and passed through a pre-trained BERT model to obtain contextual embeddings for each token. Similarity Calculation: For each token in the generated text, the cosine similarity is computed with every token in the reference text. This process is also performed in reverse from reference to generated Precision: For each token in the generated text, identify the reference token with the highest similarity. The average of these maximum similarities across all generated tokens constitutes the precision. Recall: Similarly, for each token in the reference text, find the generated token with the highest similarity. The average of these maximum similarities across all reference tokens constitutes the recall. F1 Score Calculation: The harmonic mean of precision and recall is computed to obtain the F1 score, providing a balanced measure of the model's performance. Let: $c = c_1, c_2, ..., c_3$ be the set of tokens in the generated text. r = $r_1, r_2, ..., r_n$ be the set of tokens in the reference text. E(c_i)$andE(r_j)$ denote the BERT embeddings of tokens c_i and r_j, respectively. cos_sim(x, y) represents the cosine similarity between vectors x and y.

1. Precision Equation

$$P = \frac{1}{n}\sum_{i=1}^{n} \max_{j\in\{1,\ldots,m\}} \cos\_sim(E(c_i), E(r_j))$$

2. Recall Equation

$$R = \frac{1}{m}\sum_{j=1}^{m} \max_{i\in\{1,\ldots,n\}} \cos\_sim(E(r_j), E(c_i))$$

3. F1 Score Equation

$$F_1 = \frac{2 \cdot P \cdot R}{P + R}$$

I have evaluated the Fine-Tuned SLMs on 1160 testing samples which were separated out and were not used during the Fine-tuning process.

All three SLMs demonstrated improvements across most evaluation metrics after fine-tuning on the scientific QA dataset, underscoring the general applicability of the QLoRA-based approach for adapting these models to this specialized task.The degree of these advancements differed, though. Compared to Pythia 1.4B, Phi 1.5 and TinyLlama 1.1B showed more noticeable improvements, especially in BLEU, ROUGE, and BERT F1 scores. In contrast to Pythia 1.4B, which had a relatively modest gain of +0.0312, Phi 1.5's BERT F1 score increased by +0.0736, TinyLlama 1.1B's by +0.0596. Interestingly, out of the three fine-tuned models on this dataset, the fine-tuned TinyLlama 1.1B obtained the greatest absolute BERT F1 score (0.6299) and ROUGE-L score (0.2649), even though it was the model with the fewest parameters.

For TinyLlama 1.1B there was an increase in BERT Recall (from 0.5861 to 0.6568) and a similar pattern was seen for Phi 1.5 (from 0.4711 to 0.6553). This implies that the models' capacity to recognize and extract pertinent information from the contexta critical skill for answering questionswas especially improved by the fine-tuning process. Even with the same methodology and dataset, these SLMs' varying susceptibilities to fine-tuning suggest that the base models' intrinsic qualitiessuch as their pre-training data, architectural subtleties, or pre-training objectivesplay a major part in their adaptability. For instance, TinyLlama 1.1B's impressive performance may be due to its recent and intensive pre-training routine, which may have made it extremely responsive to the new domain-specific data. The relatively modest gains of Pythia 1.4B may indicate that it would benefit from different fine-tuning hyperparameters or that its pre-training was less in line with the linguistic style or reasoning processes common in the scientific QA dataset. This variability underscores the importance of careful base model selection in PEFT strategies.

The observed variation in fine-tuning effectiveness across scientific domain namely Physics, Chemistry, and Biology reveals important insights into the interplay between model architecture, pretraining, and task specificity in low-parameter adaptation of small language models (SLMs). Across all three models examined, Phi 1.5 (1.3B), Pythia 1.4B, and TinyLlama 1.1 fine-tuning led to measurable improvements in standard evaluation metrics, including BLEU, ROUGE, and BERTScore. However, the extent and nature of these improvements were not uniform across subject areas.

Physics emerged as the domain with the highest performance gains for all three models. For instance, the fine-tuned TinyLlama achieved a BERT F1 score of 0.6315 in Physics, compared to 0.6274 in Chemistry and 0.6308 in Biology. Similarly, Phi 1.5 recorded a peak BERT F1 of 0.6020 in Physics, indicating that Physics questions were more amenable to fine-tuning under the QLoRA framework. One potential explanation is that Physics QA pairs often involve well-structured factual content, enabling more direct pattern learning and contextual grounding during the fine-tuning process. In contrast, Chemistry questions, often more symbolic or formulaic, resulted in slightly lower BERT F1 scores across all models, suggesting a greater challenge in generalization due to the abstract and syntactic nature of chemical representations. Biology, characterized by its more descriptive and natural language-rich content, occupied a middle ground in performance, potentially benefiting from pre training corpora that include biological or biomedical text.

Among the models, TinyLlama 1.1B consistently demonstrated the most substantial gains across all domains despite its smaller parameter count. This outcome underscores the importance of recent pre-training regimens and domain coverage, suggesting that smaller models with more diverse or domain-aligned pretraining can outperform larger counterparts when adapted with parameter-efficient fine-tuning techniques. Phi 1.5 also displayed consistent improvements, albeit more modest, reflecting its robustness and generalizability across scientific content. Pythia 1.4B, however, exhibited the least responsiveness to fine-tuning, with comparatively limited gains in BERT F1 and ROUGE scores. These results point to architectural and pretraining differences that influence how effectively models internalize new domain-specific information.

Overall, the findings suggest that while QLoRA enables effective adaptation of SLMs across scientific domains, the relative gains are modulated by subject characteristics and model-specific factors. Physics appears to benefit the most from this adaptation strategy, possibly due to its structured nature. Additionally, model selection remains a critical factor, as evidenced by TinyLlama's superior performance, highlighting the need for alignment between pretraining and fine-tuning distributions to maximize domain adaptation efficacy.

Table 2 Comparative Performance of Fine-Tuned SLMs on Scientific QA Dataset

Model	Status	BLEU	ROUGE-1	ROUGE-2	ROUGE-L	BERT-P	BERT-R	BERT-F1 Score	Δ(FT-Parent) BERT-F1 Score
Phi 1.5 (1.3B)	Parent	0.0027	0.186	0.118	0.152	0.61	0.4711	0.5266	0.0736
Phi 1.5 (1.3B)	Fine-tuned	0.132	0.296	0.159	0.235	0.57	0.6553	0.6002	
Pythia 1.4B	Parent	0.0347	0.2121	0.0491	0.1861	0.5481	0.4814	0.5091	0.0312
Pythia 1.4B	Fine-tuned	0.036	0.2208	0.0772	0.1929	0.5678	0.5241	0.5403	
TinyLlama 1.1B	Parent	0.065	0.3122	0.1195	0.2096	0.5655	0.5861	0.5703	0.0596
TinyLlama 1.1B	Fine-tuned	0.1028	0.3569	0.1628	0.2649	0.6236	0.6568	0.6299	

Table 3 Comparative Performance of Fine-Tuned SLMs by Subject-wise

Model	Subject	BLEU	ROUGE-L	BERT-F1 Score
Phi 1.5 (1.3B)	Physics	0.1341	0.2362	0.602
	Chemistry	0.1293	0.2347	0.5989
	Biology	0.1327	0.2341	0.5997
Pythia 1.4B	Physics	0.037	0.1943	0.5416
	Chemistry	0.0354	0.1926	0.5382
	Biology	0.355	0.192	0.5411
TinyLlama 1.1B	Physics	0.1049	0.2662	0.6315
	Chemistry	0.1008	0.2629	0.6274
	Biology	0.1026	0.3566	0.7058

Conclusion

The study demonstrated that fine-tuning of Phi 1.5, Pythia 1.4B, and TinyLlama 1.1B using QLoRA on a scientific QA dataset demonstrated notable improvements in performance metrics, particularly for Phi 1.5 and TinyLlama 1.1B. The learning curves and implemented regularization techniques (PEFT, LoRA dropout) indicate that the training was effective and overfitting was largely controlled. The clarification regarding the use of QLoRAwhich integrates LoRA with base model quantization-highlights the resource-efficient nature of the approach, enabling the adaptation of billion-parameter models on consumer-grade hardware.

Limitations, such as the lack of evaluation of the fine-tuned models on a second standard dataset, and the current inability to provide a deep granular breakdown of performance by specific scientific concepts, are acknowledged. These areas present valuable avenues for future research, which could further refine our understanding of how to best optimize and deploy SLMs for diverse scientific applications. The current work contributes by demonstrating a practical and effective methodology for enhancing SLM capabilities in the domain of scientific question answering within resource-constrained environments.

Acknowledgement

I would like to express my sincere thanks to Mrs Geeta Gehani for her guidance, encouragement, and direction in the course of this research. Her expertise and inspiration were instrumental in the direction of this work. I would also like to express my sincere thanks to my other friends and relations for their unwavering encouragement and inspiration in the course of this. Finally, I also acknowledge the National High School Journal of Science (NHSJS) for giving me this chance to have my research published and for creating a platform that celebrates students researching science and new ideas.

References

1 N. Alabbasi, O. Erak, O. Alhussein, I. Lotfi, S. Muhaidat and M. Debbah, *IEEE Internet of Things Journal*, 2025.

2 A. R. Nair, D. Gupta and B. Premjith, *Scientific Reports*, 2024, **14**, 24202.

3 S. S. Alahmari, L. O. Hall, P. R. Mouton and D. B. Goldgof, *IEEE Access*, 2024.

4 J. Devlin, M. W. Chang, K. Lee and K. Toutanova, Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), 2019, pp. 4171–4186.

5 J. Howard and S. Ruder, *Universal language model fine-tuning for text classification*, 2018.

6 N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. De Laroussilhe, A. Gesmundo and S. Gelly, International Conference on Machine Learning (ICML), 2019, pp. 2790–2799.

7 J. Pfeiffer, A. Rckl, C. Poth, A. Kamath, I. Vuli, S. Ruder and I. Gurevych, *AdapterHub: A framework for adapting transformers*, 2020.

8 X. L. Li and P. Liang, *Prefix-tuning: Optimizing continuous prompts for generation*, 2021.

9 E. Ben Zaken, Y. Goldberg and S. Ravfogel, Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), 2022, pp. 1–9.

10 E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang and W. Chen, International Conference on Learning Representations (ICLR), 2022, p. 3.

11 T. Dettmers, A. Pagnoni, A. Holtzman and L. Zettlemoyer, *Advances in Neural Information Processing Systems*, 2023, **36**, 10088–10115.

12 D. Wang, D. Kim, B. Jin, X. Zhao, T. Fu, S. Yang and X. Y. L. Yanglet, *FinLoRA: Finetuning Quantized Financial Large Language Models Using Low-Rank Adaptation on GPUs*, 2025.

13 Y. Li, S. Bubeck, R. Eldan, A. Del Giorno, S. Gunasekar and Y. T. Lee, *arXiv preprint arXiv:2309.05463*, 2023.

14 L. Chen, F. Yuan, J. Yang, X. He, C. Li and M. Yang, *IEEE Transactions on Knowledge and Data Engineering*, 2021, **35**, 3239–3252.

15 A. Patle, D. Patil, P. Patel, V. Kulkarni and A. Munshi, IEEE International Conference on Computer Vision and Machine Intelligence (CVMI), 2024, pp. 1–6.