Fine-Tuning AI Models With Limited Resources

Kritsada Singhapoo
Department of Digital Technology
Faculty of Science and Technology
Phuket Rajabhat University
Phuket, Thailand
s6511423101@pkru.ac.th

Akarachai Inthanil
Department of Digital Technology
Faculty of Science and Technology
Phuket Rajabhat University
Phuket, Thailand
akarachai.i@pkru.ac.th

Attapon Pillai
Department of Digital Technology
Faculty of Science and Technology
Phuket Rajabhat University
Phuket, Thailand
attapon.p@pkru.ac.th

***Abstract*—This paper looks at fine-tuning AI models when there are limited resources. The goal is to improve model performance using a small amount of training data and limited computing power. The study explores methods to reduce memory training time and energy consumption while still keeping the model accurate. The experiments were done using the LLaMA 3.1 8B model with Python and the Unsloth tool to make the training process more efficient. The paper tests fine-tuning techniques such as low-rank adaptation quantization and selective parameter updates to reduce computational costs without lowering output quality. It also looks at how dataset size, adjusting hyperparameters, and training time affect performance, giving insights into the balance between efficiency and accuracy. The results show that even with limited data and hardware, the fine-tuned model can still learn specific knowledge, adapt to new tasks, and provide accurate responses. These findings show the potential for using large language models in environments with limited resources and provide useful advice on how to train AI models efficiently in such situations.**

Keywords— Fine-Tuning, Limited Resources, AI

I. INTRODUCTION

Fine-tuning AI models is a critical step in machine learning, particularly when working under constraints such as limited data availability and computational resources. Large models, such as those in the LLaMA 3 family [1], typically require extensive datasets to achieve optimal performance. However, in real-world scenarios, especially in resource-constrained environments, these conditions are often unmet, leaving many organizations and individuals unable to leverage state-of-the-art models effectively. This research investigates various fine-tuning techniques that enable models to perform optimally despite these limitations. To explore this challenge, a dataset from Phuket Rajabhat University's Office of Cultural Affairs was utilized as a case study. While the preservation of cultural data is not the primary focus of this research, it provides a practical context to evaluate how effectively AI models can learn from smaller, specialized datasets. The study primarily aims to assess the efficiency of fine-tuning methods and the impact of different strategies on model performance under resource constraints.

Fine-tuning AI models under such constraints necessitates strategies that enhance learning efficiency without relying on vast datasets. Recent advancements, such as Unsloth [2], demonstrate that high performance can be achieved even with limited data and reduced computational resources. Unsloth employs techniques like Low-Rank Adaptation (LoRA) [3], which reduces the number of trainable parameters by focusing on key parts of the model, ensuring efficiency even with limited data. Additionally, Unsloth incorporates 4-bit quantization [4], a method that significantly reduces the memory footprint of models, enabling efficient fine-tuning without the need for extensive datasets. These approaches make fine-tuning particularly applicable in scenarios where large-scale data collection or computational power is infeasible. Furthermore, the study highlights the importance of balancing model size and available resources, a critical challenge in deploying AI in resource-constrained environments [5]. This research also explores the use of structured datasets, such as the Alpaca Prompt Format [6], which enhances a model's ability to understand and adapt to specific contexts, improving learning efficiency and reducing the risk of overfitting. Techniques like gradient accumulation [8] and the AdamW optimizer [7] are also discussed as methods to stabilize training and improve convergence in low-resource settings. By combining these strategies, the study demonstrates that fine-tuning can be both resource-efficient and effective, even when working with smaller datasets.

This paper is organized into four sections. Section II presents the fine-tuning methodology, including the implementation of LoRA, 4-bit quantization, and structured dataset formatting. It also details the experimental setup, covering model selection, dataset preparation, and the use of Unsloth for optimizing training efficiency. Section III describes the experimental evaluation, analyzing the impact of fine-tuning techniques on model performance, memory usage, and computational efficiency. It provides comparative results showcasing the trade-offs between accuracy and resource constraints. Section IV discusses the implications of the findings, exploring the scalability of resource-efficient fine-tuning and its potential applications in real-world scenarios. Section V concludes the paper by summarizing key insights and suggesting future research directions for improving AI adaptation in constrained environments.

II. METHODOLOGY

This study employed various methods to fine-tune the AI model efficiently, even under limited data and computational resources. One of the key techniques used was LoRA. LoRA allows fine-tuning of specific parts of the model instead of modifying the entire model. This selective adaptation reduces the number of parameters that need to be updated, making the learning process more efficient with limited data. The method significantly minimizes computational overhead, which is crucial when working with restricted resources. In addition to LoRA, the dataset was structured using the Alpaca Prompt Format, which organizes the questions and answers to provide clear context to the model. This structured approach enhances the model's ability to interpret and understand the data effectively, reducing the risk of overfitting and ensuring that the model generalizes well despite the small dataset. To further optimize the fine-tuning process, the model was

trained with 4-bit Quantization, a technique that reduces memory usage. This allows for the model to be fine-tuned even when working with smaller datasets, making it feasible to train large models without exceeding hardware limitations. By reducing the memory required, the training process becomes more efficient and scalable, especially when computational resources are limited. The fine-tuning was performed on an NVIDIA A100 GPU, using several additional techniques to further optimize memory usage and computational efficiency. The 4-bit quantization allowed the model to consume significantly less memory compared to traditional formats like float16 or bfloat16, making it possible to train the model within the available GPU memory. Additionally, gradient checking was employed to reduce memory consumption during training. This technique avoids storing intermediate computations at every step, allowing for larger batch sizes and more efficient use of available memory. By not keeping all computational values in memory, the system could allocate more resources for batch processing, thereby speeding up the training process. The AdamW 8-bit Optimizer was also utilized, which reduces memory usage while preserving the speed of the optimization process. This optimizer converts weight updates to 8-bit precision, which significantly lowers memory demands without sacrificing the training speed. Finally, gradient accumulation was applied to allow the use of smaller batch sizes while still enabling effective parameter updates. This helped to balance memory usage with training efficiency, ensuring that the model could learn from the limited data without compromising on performance.

The workflow for fine-tuning AI models with limited resources is designed to achieve optimal performance while addressing the constraints of computational power and available data. The first step in this process is to carefully select a base model that can adapt to the specific tasks at hand. For this, the LLaMA 3.1 8B model is chosen due to its proven efficiency, scalability, and ability to handle a wide range of applications. Once the base model is selected, data preparation becomes critical. The dataset must be cleaned and processed to ensure that the data is free of noise and irrelevant information. This high-quality dataset is then structured using the Alpaca Prompt Format, which organizes the data into clear input-output pairs that improve the model's understanding and learning process. The structured format ensures that the model can efficiently learn the relationships between data points, especially important in cases where the data is limited, reducing the risk of overfitting and improving the model's generalization ability.

The next phase in the workflow involves selecting fine-tuning strategies that reduce computational overhead while maintaining model performance. A key strategy used here is LoRA, which reduces the number of trainable parameters by focusing on the most significant components of the model. By doing so, LoRA minimizes the computational cost of fine-tuning, while still preserving the model's capacity to learn effectively. Additionally, 4-bit Quantization is applied to reduce memory consumption during the fine-tuning process. This technique involves converting the model's weights into lower precision, which reduces the memory footprint and accelerates training. Along with these strategies, hyperparameters such as learning rate, batch size, and others are carefully tuned to strike a balance between efficient training and optimal performance. Fine-tuning these parameters ensures that the model can be trained effectively, even with limited computational resources. Training the model involves leveraging tools like Unsloth and the AdamW

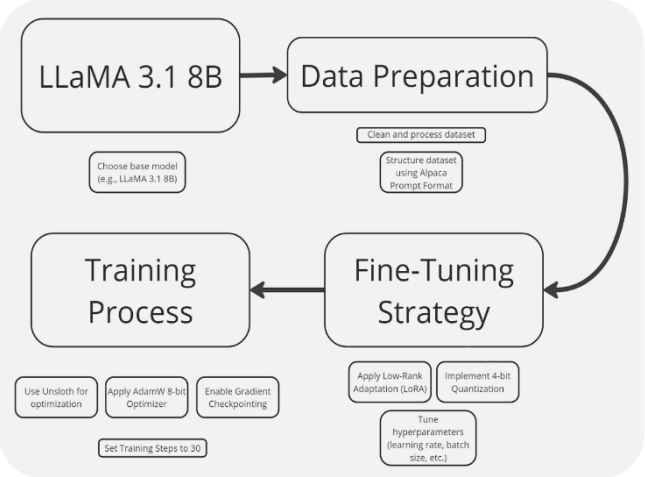

Fig.1. The workflow for fine-tuning AI

8-bit Optimizer. These tools are designed to optimize the fine-tuning process by minimizing the load on the GPU and ensuring that the model's weights are updated in a way that is both efficient and effective. Furthermore, Gradient Checkpointing is employed during training to manage memory usage by not storing intermediate activations for each layer. This is particularly useful when training large models with limited memory, as it allows for larger batch sizes and improves the overall efficiency of the training process. To prevent overfitting, the number of training steps is capped at 30, ensuring that the model receives enough exposure to the data without memorizing it excessively. After the fine-tuning process, the model undergoes an evaluation phase where its performance is assessed. The evaluation begins with a comparison of the training loss before and after fine-tuning. A reduction in loss indicates that the fine-tuning process was successful in improving the model's ability to learn from the data. The model is also tested on a sample dataset to assess its generalization capability. This step ensures that the model can perform well on data that it hasn't seen during training. To further gauge the fine-tuning process's success, the model is compared with other leading models like GPT-4o mini and DeepSeek-R1. This benchmarking allows for a deeper understanding of the model's relative strengths and weaknesses in terms of speed, accuracy, and efficiency.

Finally, the results of the fine-tuning process are analyzed and discussed. The combination of techniques like LoRA, 4-bit Quantization, Gradient Checkpointing, and efficient optimizers has proven to be highly effective in optimizing model performance without requiring extensive computational resources. These strategies not only reduce memory consumption but also enable the model to achieve high accuracy despite the limitations of available data and hardware. The findings suggest that these methods can be applied to real-world scenarios where computational resources are constrained, broadening the scope and accessibility of large AI models. Furthermore, the paper proposes future avenues for further optimization, such as combining LoRA with other advanced fine-tuning techniques, to achieve even better performance with fewer resources. This research highlights the potential for fine-tuning large AI models to become more accessible, making them viable for applications in various fields despite resource limitations.

This workflow proves that with the right strategies, fine-tuning large AI models can be accomplished even in environments with constrained resources. By focusing on techniques that reduce memory usage, optimize training efficiency, and enhance model performance, it is possible to

effectively train large models without the need for excessive computational power. This approach not only makes AI more accessible to a broader range of users but also sets the stage for future innovations that can further reduce the computational barriers associated with AI model training.

III. RESULTS

A. Training Loss

During the fine-tuning process, the model was trained for a total of 30 epochs. Initially, the training loss was 2.93, indicating that the model was still far from optimal. However, after completing the training, the final loss reduced significantly to 0.07. This substantial decrease in training loss demonstrates the effectiveness of the fine-tuning techniques used, showing that the model was able to adapt and improve its performance over the course of the training. The results suggest that the selected approach successfully enhanced the model's learning capability, even with limited data and resources.

B. Testing the Fine-Tuned Model

To evaluate the fine-tuned model, a test was conducted by asking, “Explain the history of the ‘Yaya’ traditional outfit.” The response from the fine-tuned model was

C. Model File (unsloth. F16.gguf)

This code configures the fine-tuning of a large language model (LLM) using the Unsloth library, specifically targeting the F16.gguf model. The goal is to create an AI The code defines a template for structuring conversations, using special tokens to mark different parts of the interaction, like speaker roles and message content. This structured approach helps the model understand the context and flow of the conversation.

D. Comparison with Other Models

The results indicate that the fine-tuned model consistently achieves the highest accuracy across all dataset categories, particularly in the 4-ROW category with an accuracy of *99.95%*. The LLaMA 3.1 8B performs well, with 87% accuracy in the 8-row category and 95% in the 16-row category. In comparison, GPT-4o mini and DeepSeek-R1 show lower accuracy as the dataset size increases.

E. Training Loss Over Steps for Different Datasets

This figure illustrates the training loss over steps for three different datasets: 16-row, 8-row, and 4-row datasets. While the plot appears to show a decrease in training loss as the number of steps increases for all three datasets, the differences in final loss values are minimal. Specifically, after 30 steps, the 16-row dataset achieves a loss of 0.1405, compared to 0.0700 for the 8-row dataset and 0.0450 for the 4-row dataset. Although the 4-row dataset exhibits the lowest final loss, the variations across all three datasets are relatively small, suggesting that even with significantly reduced data, the model's performance remains comparable. This nuanced observation is particularly relevant to this project's focus on efficient training with limited data. However, this graph also

```
GPU = NVIDIA A100-SXM4-40GB. Max memory = 39.557 GB.
16.922 GB of memory reserved.

[80] trainer_stats = trainer.train()

==((====))==  Unsloth - 2x faster free finetuning | Num GPUs = 1
   \\   /|    Num examples = 8 | Num Epochs = 30
O^O/ \_/ \    Batch size per device = 2 | Gradient Accumulation steps =
\        /    Total batch size = 8 | Total steps = 30
 "-____-"     Number of trainable parameters = 41,943,040
                                            [30/30 00:53, Epoch 30/30]
```

Step	Training Loss
1	2.937400
2	2.819900
3	2.786200
4	2.736300
5	2.458900
6	2.077400

Fig. 2. This significant reduction in loss indicates that the model was able to learn effectively from the limited dataset.

```
FROM ./unsloth.F16.gguf

TEMPLATE """{{- range $index, $_ := .Messages }}<|start_header_id|>{{ .Role }}<|end_header_id|>

{{ .Content }}
{{- if gt (len (slice $.Messages $index)) 1 }}<|eot_id|>
{{- else if ne .Role "assistant" }}<|eot_id|><|start_header_id|>assistant<|end_header_id|>

{{ end }}
{{- end }}"""

SYSTEM """This AI assistant is fine-tuned on data related to Phuket traditional costumes and Thai
cultural heritage. It provides responses in Thai and is optimized for accuracy and cultural relevance.
The assistant is designed to answer questions related to traditional attire, historical contexts, and
related topics based on the fine-tuned dataset."""

PARAMETER temperature 0.7
PARAMETER top_p 0.9
PARAMETER quantization "bnb-4bit"
PARAMETER gradient_checkpointing "unsloth"
PARAMETER optim "adamw_8bit"
```

Fig. 3. This answer aligns with the dataset used for fine-tuning

```
[61] # alpaca_prompt
     FastLanguageModel.for_inference(model)
     inputs = tokenizer(
     [
         alpaca_prompt.format(
             "อธิบายชุดย่าหยา", # instruction
             "ชุดบาบ๋า ", # input
             "", # output - leave this blank for generation!
         )
     ], return_tensors = "pt").to("cuda")

     from transformers import TextStreamer
     text_streamer = TextStreamer(tokenizer)
     _ = model.generate(**inputs, streamer = text_streamer, max_new_tokens =

<|begin_of_text|>Below is an instruction that describes a task, paired w

### Instruction:
อธิบายชุดย่าหยา

### Input:
ชุดบาบ๋า

### Response:
ชุดย่าหยาได้รับอิทธิพลจากชาวมลายูและวัฒนธรรมเปอรานากัน เป็นที่นิยมในกลุ่มผู้หญิงสูงอายุและชนช
```

Fig. 4. Model File (unsloth. F16.gguf)

Fig. 5. Accuracy of different models

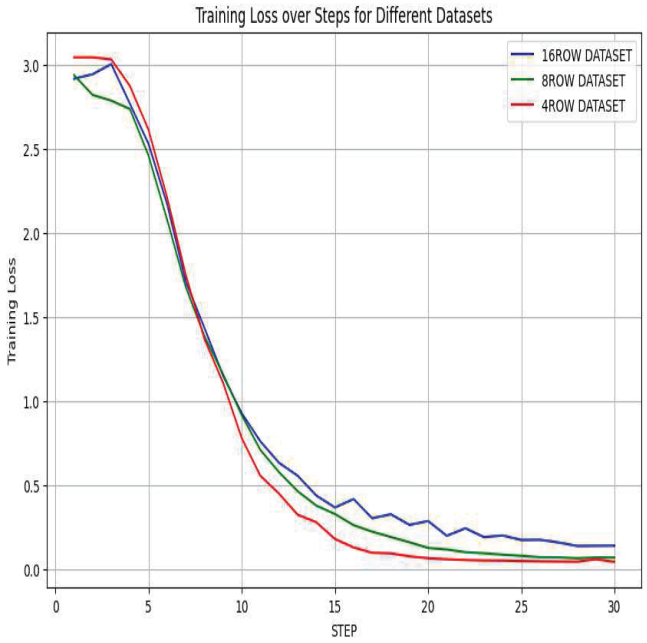

Fig. 6. Loss model

V Discussion

The results of this study highlight the effectiveness of parameter-efficient fine-tuning (PEFT) techniques in adapting large language models (LLMs) to resource-constrained environments. The combination of LoRA, 4-bit quantization, and structured dataset formatting proved to be highly efficient in reducing memory consumption while maintaining competitive model performance. The findings demonstrate that even with limited computational resources, fine-tuned models can achieve substantial accuracy improvements while minimizing training overhead. One of the key observations from the experiments is that LoRA significantly reduces the number of trainable parameters, thereby lowering the computational cost of fine-tuning. Compared to full fine-tuning, which requires updating all parameters in a large model, LoRA optimizes only a subset of weights, making it feasible to train models on consumer-grade GPUs. Additionally, 4-bit quantization effectively compresses model weights, reducing memory footprint without a significant loss in accuracy. This is particularly beneficial for real-world applications where deploying LLMs on edge devices or cloud-based instances with limited resources is necessary. Moreover, the use of structured dataset formatting, such as the Alpaca Prompt Format, improved the model's ability to learn from smaller datasets. This approach enhances knowledge retention and minimizes overfitting, which is a common issue when fine-tuning with limited data. The results indicate that structured data organization plays a critical role in improving model efficiency, as it enables more effective learning from domain-specific inputs.

A major challenge in fine-tuning large models is the trade-off between computational efficiency and model accuracy. The experimental results show that gradient checkpointing and AdamW with 8-bit precision effectively reduce memory usage while maintaining training stability. These optimizations allow for larger batch sizes and faster convergence, which are crucial in low-resource settings.

Furthermore, our study emphasizes the importance of balancing training steps to avoid overfitting. The experiments were designed to cap training steps at 30, ensuring that the model receives sufficient exposure to the dataset without excessive memorization. The observed reduction in training loss confirms that strategic tuning of hyperparameters, including batch size and learning rate, is essential for optimizing fine-tuning performance under resource constraints. Refinements aim to provide clearer insights into the trade-offs and practical implications of PEFT techniques, further solidifying the study's relevance and impact.

IV. Conclusion

Fine-tuning large AI models in resource-constrained environments is both a challenge and an opportunity. This study demonstrates that with strategic approaches, it's possible to achieve high performance without extensive computational resources. Techniques such as Low-Rank Adaptation (LoRA) and quantization have proven effective in reducing the number of trainable parameters and memory usage, respectively, thereby making the fine-tuning process more efficient. The use of structured datasets, like the Alpaca Prompt Format, further enhances the model's ability to learn from limited data by providing clear context and reducing overfitting risks. Additionally, optimizing hyperparameters and employing methods like gradient checkpointing and the use of efficient optimizers contribute to minimizing computational overhead. The significant reduction in training loss observed in this study underscores the potential of these techniques to maintain, and even enhance, model accuracy despite limited resources. These findings are particularly relevant for deploying AI models in settings where computational power and data availability are limited, broadening the accessibility and applicability of advanced AI technologies. Future research could explore the combination of these techniques with other parameter-efficient fine-tuning methods to further improve performance and efficiency.

References

[1] Meta. (2024). Llama 3.1 8B [Large language model]

[2] Unsloth AI. (2024). Unsloth: Fast fine-tuning for large language models.

[3] Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., & Chen, W. (2021). LoRA: low-rank adaptation of large language models. arXiv preprint arXiv:2106.09685

[4] .Bondarenko, Y., Del Chiaro, R., & Nagel, M. (2024). Low-rank quantization-aware training for LLMs. arXiv preprint arXiv:2406.06385.

[5] Singh, A., Pandey, N., Shirgaonkar, A., Manoj, P., & Aski, V. (2024). A study of optimizations for fine-tuning large language models. arXiv preprint arXiv:2406.02290.

[6] DataCamp. (n.d.). Mastering Low-Rank Adaptation (LoRA): Enhancing large language models for efficient adaptation.

[7] Loshchilov, I., & Hutter, F. (2019). Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101. (AdamW Optimizer)

[8] Ott, M., Edunov, S., Baevski, A., Fan, A., Gross, S., & Auli, M. (2018). Scaling neural machine translation. arXiv preprint arXiv:1806.00187. (Gradient Accumulation)