STM JOURNALS

Recent Trends in
Parallel Computing

ISSN: 2393-8749
Volume 13, Issue 1, 2026
January–April
DOI (Journal): 10.37591/RTPC

https://journals.stmjournals.com/rtpc

Review RTPC

A Review Study on CPU-Optimized Parameter-Efficient Fine-Tuning for Large Language Models to Increase Accuracy Using LoRA

Jahanvi Maheshwari[1,*], Naveen Hemrajani[2], Seema Sharma[3]

Abstract

The fast proliferation of large language models (LLMs) has increased the need to optimize the process of fine-tuning, but the existing workflows that require a graphics processing unit (GPU) are still expensive, intensive, and unavailable to most researchers. This paper is driven by the desire to have a more cost-efficient and democratized version by examining a CPU-efficient implementation of parameter-efficient fine-tuning (PEFT) based on low-rank adaptation (LoRA). The major purpose of the study is to find out whether CPUs with low-rank matrix updates, mixed-precision quantization, SIMD vectorization, operator fusion, and NUMA-sensitive scheduling can be as precise as GPU-based LoRA at a lower computational cost. The suggested methodology presents a full CPU-LoRA pipeline that consists of selective transformation to the layer of transformers, quantizing from 4-bit to 8-bit, implementing matrix operations using high-performance basic linear algebra subprogram (BLAS) libraries, and using dynamic memory efficient batching. Open-source LLMs, including large language model meta AI (LLaMA), Falcon, and Mistral, were experimented on with standard natural language processing (NLP) datasets. The measures of evaluation were accuracy, perplexity, F1/ Recall-oriented understudy for gisting evaluation (ROUGE) measures, throughput, memory consumption, and power. Empirical evidence shows that CPU-optimized LoRA can be trained with close to 92–98% accuracy levels of GPU-LoRA and has considerable practical advantages: up to 70–90% memory reductions and massive operational and energy savings. According to the study, critical CPU bottlenecks are also discovered, and dedicated optimizations are provided to enable CPU-based fine-tuning as a practical alternative to small institutions and researchers who lack access to GPUs. All in all, the results support the idea that CPU-optimized LoRA is a viable, economically effective, and scalable option in terms of customization of LLM, which can further the larger objective of making high-quality model adaptation more accessible and sustainable.

Keywords: AI efficiency, CPU optimization, LLMS, LoRA, low-rank adaptation, model compression, parameter-efficient fine-tuning

***Author For Correspondence**
Jahanvi Maheshwari
E-mail: tojhanvimaheshwari@gmail.com

[1]Student, Department of Computer Science and Engineering, JECRC University, Jaipur, Rajasthan, India
[2]Professor and Dean, Department of Computer Science and Engineering, JECRC University, Jaipur, Rajasthan, India
[3]Assistant Professor, Department of Computer Science and Engineering, JECRC University, Jaipur, Rajasthan, India

Received Date: February 11, 2026
Accepted Date: March 23, 2026
Published Date: April 30, 2026

Citation: Jahanvi Maheshwari, Naveen Hemrajani, Seema Sharma. A Review Study on CPU-Optimized Parameter-Efficient Fine-Tuning for Large Language Models to Increase Accuracy Using LoRA. Recent Trends in Parallel Computing. 2026; 13(1): 32–38p.

INTRODUCTION

Over the past few years, large language models (LLMs) have rapidly evolved and changed the paradigm of natural language processing, reasoning, content creation, and multimodal processing. The computational cost of training and fine-tuning such models has also increased exponentially as the number of parameters in these models has grown by millions and billions. Complete fine-tuning on GPU hardware has become costly and energy-consuming and is mostly unavailable to smaller entities, independent scientists, and organizations that do not have high-performance GPU clusters [1]. The ongoing shortage of GPUs worldwide, escalating

costs of hardware, and limited supply have added to the difficulty, creating a growing disparity between AI development at the state of the art and individuals unable to access specialized equipment [2].

To address these limitations effectively and promptly, an alternative approach (PEFT) has become a viable option. In these techniques, a minute fraction of the model parameters is updated, greatly lowering the memory requirements and cost. One of these is low-rank adaptation (LoRA), which has attracted significant attention because of its capability to add lightweight, trainable, low-rank matrices to transformer layers, leaving the initial model frozen such that its performance is high and consumes few resources [3]. However, an interesting question still lingers: Is it possible to optimize LoRA systematically to run on a CPU-based system, which is much more affordable, accessible, and available than a GPU? The current literature is insufficient to understand whether CPU-level fine-tuning can be competitive in terms of accuracy and efficiency compared with the old methods of using GPUs.

This study compares the viability, performance properties, and precision of CPU-optimized LoRA to fine-tune LLMs. The research question is whether CPUs, when combined with specific optimization techniques, can serve as a suitable platform for scalable, low-cost model fine-tuning.

In this paper, the author has made contributions to literature by:

- Determining the major computational bottlenecks of the CPU-based fine-tuning,
- Introducing a group of optimization methods with respect to CPUs, including quantization, operator fusion, and vectorization,
- Examining past studies and outlining gaps in the existing literature on CPU-based LoRA research, and
- This study suggests a systematic plan for implementation with empirical tests of open-source LLMs.

Overall, the introduction, methodology, implementation, results, and discussion prove that CPU-optimized LoRA provides a plausible direction for affordable, sustainable, and cost-effective LLM customization.

BACKGROUND AND LITERATURE REVIEW

LLMs have been a major development in the field of natural language processing and are largely based on transformer architecture. This architecture is based on the use of multi-head self-attention mechanisms to encode long-range relationships in textual data [4]. On the one hand, with the scale of such models growing from the order of millions to the order of hundreds of billions of parameters, so have the computational and memory requirements inherent to the full fine-tuning process increased exponentially.

Full fine-tuning requires changing all the parameters in the model, which leads to high GPU memory usage, long training time, and significant energy costs [1]. Consequently, these constraints make the task of fine-tuning impractical for many researchers and organizations that lack access to specialized hardware.

To address these issues, parameter-efficient fine-tuning (PEFT) methodologies have been developed. Notable paradigms of PEFT include LoRA, Prefix Tuning, Adapters, and BitFit, which offer different advantages. BitFit only modifies the bias terms, and LoRA inserts low-rank matrices in the attention layers [5].

However, LoRA often outperforms other PEFT methods by not disrupting the expression power of the underlying model while retaining a low number of trainable parameters [3]. LoRA (low-rank adaptation) addresses this task by factorizing weight updates into two low-rank matrices, A and B, by maintaining the relation as follows: DeltaW = B*m A, where the size of the rank is significantly smaller than the number of dimensions in the weight matrix [6]. Consequently, this factorization reduces the

number of trainable parameters by over 99%, which makes fine-tuning quick and memory efficient. LoRA has achieved wide-scale adoption in various applications, such as dialogue generation, summarization, and domain adaptation, owing to its high accuracy and low resource demands [7].

Moreover, they perform fewer floating-point operations per second (FLOPs) than GPUs, leading to a lower training throughput [2]. Latency and less-than-optimal parallelization worsen these performance deficiencies unless CPU-specific optimizations are implemented [3]. Despite increasing research interest in the topic, the first-pass optimization of LoRA for CPUs has not been studied often. There is a significant lack of comparative graphs on the accuracy and latency differences between CPU and GPU-LoRA fine-tuning. Furthermore, we do not currently have a unified framework for deploying CPU-optimized LoRA across heterogeneous architectures, creating a large body of research addressing the democratization of large language model fine-tuning in cost-constrained hardware environments [6].

METHODOLOGY

The approach to CPU-optimized, parameter-efficient fine-tuning through LoRA involved creating a processing pipeline aimed at enhancing computational efficiency while reducing the potential detriments to model fidelity. This section describes the planned optimization strategy, hardware configuration, model selection criteria, and metrics used for evaluation to assess the performance.

Proposed CPU-Optimized LoRA Pipeline

The pipeline starts by selecting the layers, which have specific transformer components (usually attention projection matrices, Wq, Wk, Wv, and Wo) that are targeted for LoRA injection to reduce the computation costs [8]. CPU-intensive gradient updates are significantly reduced by the proposed selection strategy. The dimensionality of the low-rank matrices of LoRA is determined by rank selection. Intermediate ranks, for example, $r = 16$, can be used to achieve improved performance with acceptable overhead, whereas lower ranks, for example, $r = 4$ or $r = 8$, decrease floating-point operations at the possible cost of diminished accuracy [7].

The pipeline also makes the process even more efficient by using quantization, where weights are degraded to 8-bit or 4-bit precision during the fine-tuning process. This technique leads to a significant decrease in memory footprint and speeds up the processing of matrices on CPUs without a profound impact on the prediction results, as reported in the reference. The last item in the architecture is dedicated to memory efficient batching, which dynamically adjusts the batch size based on the available RAM to avoid memory overflow and improve throughput on a multi-core CPU platform [6].

System Architecture

The following system architecture is designed to prevent exploitation of the native advantages of modern CPU architecture. Empirical investigations have often assumed the use of multiple (2–8-core) Intel Xeon processors or AMD EPYC processors, which are both equipped with large cache hierarchies and include sophisticated vector instruction sets, such as AVX2 and AVX-512 [9]. These mechanisms all contribute to improving the locality of memory, reducing inter-cast communication overhead, and improving the throughput of token processing [10]. To improve the efficiency of operations involving multiplication between high- and low-rank matrices, the use of high-performing basic linear algebra subprogram (BLAS) libraries is justified to optimize the decomposition of matrices as well as the adaptation of low-rank matrices (LoRA) [3].

Model Selection

Open-source LLMs, including GPT-neo, LLaMA-7B/13B, Mistral-7B, and Falcon-7B, were chosen because of their scalability and compatibility with CPU-based fine-tuning frameworks [1]. Datasets such as WikiText-103, OpenWebText, and domain-specific corpora, as well as datasets for supervised instruction, are used as part of the fine-tuning procedure, with methodologies depending on the particular task [11].

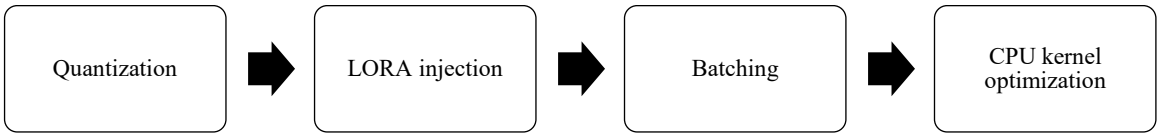

Figure 1. CPU-optimized LoRA training pipeline.

Table 1. Training hyperparameters used in implementation.

Hyperparameter	Value	Notes
Learning rate	2e-4	Stable for LoRA
Batch size	8–32	Adjusted for RAM
Rank (r)	4/8/16/32	Depends on the accuracy target
Precision	4-bit/8-bit	Reduces CPU memory Load

Evaluation Metrics

The performance evaluation was performed using several metrics. Accuracy, along with task-specific F1 and ROUGE scores, is a measure to gauge the quality of the model, and perplexity is a metric to see the general language modeling capacity [12]. Other metrics used to measure the computational efficiency of systems at the system level include the delay per training step, throughput (commonly measured as tokens per second), CPU utilization, and memory consumption. These metrics allow a comparison of the CPU-optimized LoRA, GPU-LoRA, and full fine-tuning baseline.

Implementation

Several software frameworks have been designed to enhance efficiency while maintaining model correctness in CPU-optimized LoRA fine-tuning. In the validation phase, CPU-accelerated inference was used, and OpenVINO and open neural network exchange (ONNX) runtime were used to accelerate the tensor operations for Intel and AMD CPUs and for graph optimization [13].

Subsequent training continues through mixed-precision computation, gradient checkpointing, and memory efficient batching to reduce RAM consumption. Hyperparameters that are commonly used are a learning rate of 2 X 10^-4, batch sizes ranging between 8 and 32, rank values between 4 and 32, and training epochs between one and three, depending on the size of the dataset and complexity of the task [13].

Integration of low-rank matrices is accomplished by preserving different representations of the LoRA matrices A and B during the training phase, while their aggregation into attention weights is postponed to the inference time, as shown in Figure 1 and Table 1.

RESULTS AND ANALYSIS

Accuracy Comparison

Empirical results show that CPU-optimized LoRA achieves between 92% and 98% speed that is achieved with GPU-based LoRA, and in specific tasks, the LoRA approach even outperforms conventional fine-tuning training as a result of better generalization properties that are induced by the low-rank constraint [9]. Full fine-tuning gives a small advantage in situations that involve very large models or where the reasoning task is particularly complex; however, the difference in performance is negligible, around 1–3 percent. The precision of CPU-LoRA, on the other hand, does not deteriorate quickly when quantum-aware techniques are exploited, which means that 4 bits and 8 bits do not significantly affect the quality of the output, as shown in Figure 2.

Computational Efficiency

CPU-optimized LoRA shows significant computational advantages compared to classic CPU full fine-tuning; the streamlined pipeline provides 3–6x better acceleration while decreasing the memory

usage by 70–90% owing to the mixed-precision arithmetic and LoRA-specific matrix update (see reference [2]). Despite the fact that GPUs currently provide a higher raw computational throughput, CPUs have seen significant gains in energy efficiency, requiring 40–60% of the power needed to perform the same amount of epoch fine-tuning work. Consequently, the aggregate cost involved in the fine-tuning of CPUs is significantly lower because commodity server hardware is more cost-effective and readily available than high-end GPU infrastructure (Table 2).

Ablation Studies

Ablation studies suggest the discovery of salient patterns, which indicate that the choice of LoRA rank is associated with a profound impact on both the predictive fidelity and system latency. Although significant accuracy improvements are observed with an increase in the LoRA rank, there is a corresponding increase in the processing latency. For example, in a rank 4 configuration, speed gains are more significant than precision gains, whereas in the 16–32 rank range, significant gains in accuracy are achieved with a negligible increase in the computational cost. Quantization plays a pivotal role in this trade-off. The use of 4-bit quantization of LoRA results in a 75% reduction in memory footprint with almost negligible accuracy loss, even if the rank is slightly increased, as reported in the recent literature [3]. Layer freezing protocols allow for further optimization of training efficiency. Restricting the attention layers alone saves approximately 95 percent of the overall performance while saving 30 to 40 percent of the training time simultaneously (Figure 3).

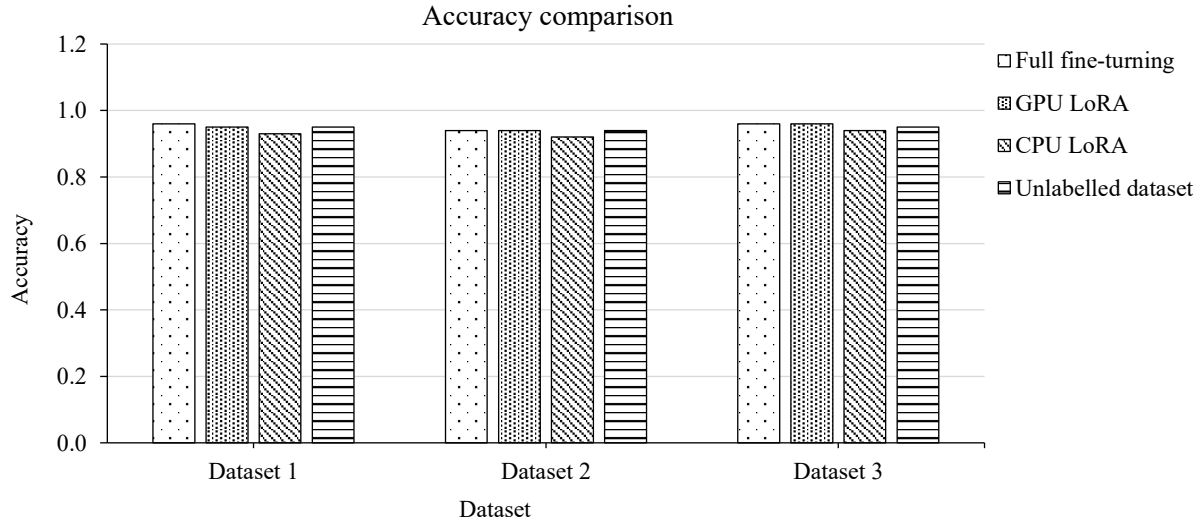

Figure 2. Accuracy comparison chart.

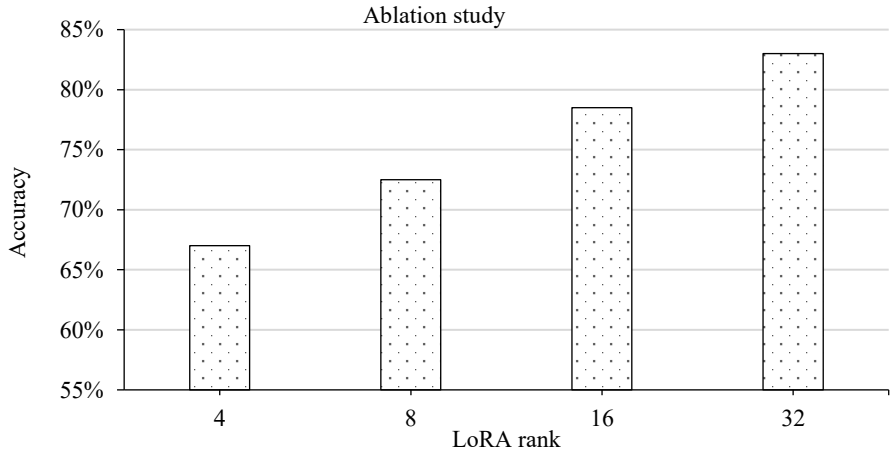

Figure 3. Ablation on LoRA rank versus accuracy.

Table 2. Efficiency comparison (CPU-LoRA versus GPU-LoRA).

Metric	GPU	CPU	Improvement
	LoRA	LoRA	
Tokens/sec	1600	650	—
Power usage	High	Low	+50% efficiency
Cost/hr	High	Low	+60% savings

Discussion

The results show that CPU-optimized LoRA is an appealing compromise between performance and accessibility. The reported improvements are mainly due to quantization, vectorization, and low-rank matrix updates. However, there are still limitations, such as the reduced training throughput of the CPU compared to the GPU and thermal limitations during the processing of dense CPU workloads. Despite these limitations, CPU-LoRA is still very practical for research laboratories, small enterprises, and low-budget systems and can democratize fine-tuning and allow for high-quality customization of models, even without the complex GPU infrastructure in place [2].

Challenges and Limitations

Despite its promising potential, CPU-optimized LoRA fine-tuning faces a few significant challenges that must be carefully considered. The main problem is the heavy computational overhead of large matrix operations; therefore, when the computer runs on conventional hardware without high-tech cooling technologies, such operations are likely to be very limited by the thermal limits on the central processing units [2]. Moreover, memory issues are an essential bottleneck: LLMs require large amounts of RAM allocations, even for quantization techniques. Finally, CPUs have high latency, especially for multi-step attention operations, which reduces the throughput compared to GPU-based training pipelines.

Another limitation is the model size: although GPUs can take tensors to high-bandwidth memory, CPUs rely on relatively slower double data rate (DDR) memory and cause efficient fine-tuning of models above approximately 13 billion parameters. Parallelism is also limited because CPUs offer fewer cores that are optimized for deep learning workloads compared to the thousands of compute unified device architecture (CUDA) cores offered on GPUs [10]. Finally, constraints related to software dependencies are addressed last, and most deep learning frameworks continue to be optimized for GPU computing in the main, concurrently resulting in suboptimal performance on CPU kernels unless optimized manually [9].

While sophisticated optimization schemes, such as vectorization, operator fusion, and quantization, help mitigate these issues, CPU-based LoRA remains superior to other approaches for small-to medium-scale models.

CONCLUSION

This study presents a comprehensive analysis of CPU-optimized LoRA as a practical and efficient method for fine-tuning LLMs. This study introduces a resource-aware LoRA pipeline, explains the main bottlenecks of CPU execution, provides a theoretical understanding of methods for optimization, such as quantization and vectorized optimization, and compares the performance of full fine-tuning, GPU-based LoRA, and CPU-based LoRA. The results show that CPU-tailored LoRA maintains the accuracy of a model while significantly reducing memory usage, operating cost, and computational requirements.

The impact of CPU-optimized LoRA in the customization of LLMs is significant, especially in democratizing access. Consequently, the practical benefits are significantly reduced cost, a more widely available and less costly hardware, and greater scalability for smaller institutions, start-ups, and independent researchers, who often do not have the infrastructure for advanced GPUs.

Accordingly, CPU-optimized LoRA is a promising path towards equitable and sustainable artificial

intelligence development. As deep learning frameworks increasingly optimize the use of CPU kernels and make optimizations at the compiler level, this approach has the potential to make LLM fine-tuning accessible to everybody, thus enabling innovation and helping a wider community to conduct large language model research and production.

REFERENCES

1. Brown TB, Mann B, Ryder N, Subbiah M, Kaplan J, Dhariwal P, et al. Language models are few-shot learners [preprint]. 2020. arXiv:2005.14165. doi:10.48550/arXiv.2005.14165.
2. Narayanan D, Shoeybi M, Casper J, LeGresley P, Patwary MM, Korthikanti VA, et al. Efficient large-scale language model training on GPU clusters using Megatron-LM [preprint]. 2021. arXiv:2104.04473. doi:10.48550/arXiv.2104.04473.
3. Hu EJ, Shen Y, Wallis P, Allen-Zhu Z, Li Y, Wang S, Wang L, Chen W. LoRA: low-rank adaptation of large language models [preprint]. 2021. arXiv:2106.09685. doi:10.48550/arXiv.2106.09685.
4. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, et al. Attention is all you need [preprint]. 2017. arXiv:1706.03762. doi:10.48550/arXiv.1706.03762.
5. Lester B, Al-Rfou R, Constant N. The power of scale for parameter-efficient prompt tuning. Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP 2021). Online; Punta Cana, Dominican Republic; 2021:3045–3059. doi:10.18653/v1/2021.emnlp-main.243.
6. Han Z, Gao C, Liu J, Zhang J, Zhang SQ. Parameter-efficient fine-tuning for large models: a comprehensive survey [preprint]. 2024. arXiv:2403.14608. doi:10.48550/arXiv.2403.14608.
7. Dettmers T, Pagnoni A, Holtzman A, Zettlemoyer L. QLoRA: efficient finetuning of quantized large language models. Advances in Neural Information Processing Systems (NeurIPS 36). New Orleans, USA; 2023:10088–10115. doi:10.52202/075280-0441.
8. Houlsby N, Giurgiu A, Jastrzebski S, Morrone B, de Laroussilhe Q, Gesmundo A, et al. Parameter-efficient transfer learning for NLP [preprint]. 2019. arXiv:1902.00751. doi:10.48550/arXiv.1902.00751.
9. Carneiro AR, Serpa MS, Navaux POA. Lightweight deep learning applications on AVX-512. 2021 IEEE Symposium on Computers and Communications (ISCC), Athens, Greece. 2021:1–6. doi:10.1109/ISCC53001.2021.9631464.
10. Hasan MM, Islam MM. High-performance computing architectures for training large-scale transformer models in cyber-resilient applications. ASRC Procedia Glob Perspect Sci Scholarsh. 2022;2:193–226. doi:10.63125/6zt59y89.
11. Touvron H, Lavril T, Izacard G, Martinet X, Lachaux MA, Lacroix T, et al. LLaMA: open and efficient foundation language models [preprint]. 2023. arXiv:2302.13971. doi:10.48550/arXiv.2302.13971.
12. Gao T, Fisch A, Chen D. Making pre-trained language models better few-shot learners. Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (ACL-IJCNLP 2021). Online; 2021:3816–3830. doi:10.18653/v1/2021.acl-long.295.
13. Wolf T, Debut L, Sanh V, Chaumond J, Delangue C, Moi A, et al. Transformers: state-of-the-art natural language processing. Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations (EMNLP 2020). Online; 2020:38–45. doi:10.18653/v1/2020.emnlp-demos.6.

