Uluslararası İleri Doğa Bilimleri ve Mühendislik Araştırmaları Dergisi
Sayı 9, S. 27-32 9, 2025
© Telif hakkı IJANSER'e aittir
Araştırma Makalesi

IJANSER
https://as-proceeding.com/index.php/ijanser
ISSN:2980-0811

International Journal of Advanced Natural Sciences and Engineering Researches
Volume 9, pp. 27-32, 9, 2025
Copyright © 2025 IJANSER
Research Article

LLaMA-2 and LLaMA-3 Models with SparseGPT-Based Model Compression and LoRA/QLoRA Integration: Efficient Large Language Model Optimization for Resource-Constrained Environments

Erke Arıbaş*[1], Melisa Güler [1]

[1]*Istanbul Technical University, Türkiye*

(aribas@itu.edu.tr) Email of the corresponding author

(Received: 01 September 2025, Accepted: 07 September2025)

(7th International Conference on Innovative Academic Studies ICIAS 2025, September 02-03, 2025)

ATIF/REFERENCE: Arıbaş, E. & Güler, M. (2025). LLaMA-2 and LLaMA-3 Models with SparseGPT-Based Model Compression and LoRA/QLoRA Integration: Efficient Large Language Model Optimization for Resource-Constrained Environments, *International Journal of Advanced Natural Sciences and Engineering Researches*, 9(9), 27-32.

Abstract – Data exists in different sizes according to the needs and usage area. The need for these storage areas is increasing day by day. However, the increasing size of these models significantly increases storage requirements, memory usage, and inference latency, limiting applicability, especially on resource-constrained devices (e.g., mobile, embedded systems). By 2025, global data volume is expected to reach 175 zettabytes, underscoring the growing need not only for efficient storage solutions but also for effective model optimization techniques [1]. Increasing model sizes and evolving computational requirements reveal the importance of model compression. The choice of compression algorithm plays a decisive role in balancing efficiency, speed, and accuracy. If the correct and efficient compression technique is chosen, both the compression ratio can be increased and the processing time can be reduced, while choosing an algorithm that is not suitable for the purpose can distract us from the target. When llama-2 and llama-3, which are among the large language models, are examined, the fact that they have millions of parameters can cause serious problems, especially in devices with limited storage and performance. The appropriate compaction technique can be selected by taking into account the characteristics as well as the overall performance metrics and size. With the SparseGPT used in this study, weights that can be considered unnecessary are pruned and the model is reduced. Using the SparseGPT-based pruning method, up to 50-60% sparsity was applied on the LLaMA-2 and LLaMA-3 models, and significant memory savings were achieved without the need for retraining by setting the insignificant weights in the parameter space of the model to zero. This approach reduces the model size while eliminating the costly requirement of full retraining. Then the model is customized with lora/qlora. Instead of completely retraining large models, additional adjustments are made through adapters.

Keywords – SparseGPT; LoRA; QLoRA; Model Compression; Large Language Models.

I. INTRODUCTION

LLaMA-2 and LLaMA-3 language models achieve strong performance across diverse tasks, but their effectiveness comes at the cost of billions of parameters [4]. This high number of parameters presents significant disadvantages to the user when using it. The main reason for this is the high memory consumption. In today's world, where resource consumption and storage are now serious concerns, various methods are used to prevent this disadvantage from outweighing the advantage. One of the most prominent of these is sparsegpt, which is one of the modern compression techniques, and the LORA/QLORA integration, which we will adapt to the result we have obtained later.

Today, the sizes of large language models reaching billions of parameters not only increase the demand for storage capacity, but also lead to a serious calculation burden and cost increase in model training, extraction and operational management processes. On the one hand, this pushes the limits of hardware resources, and on the other hand, raises sustainability issues in terms of time, energy and budget. Artificial intelligence applications today rely on models of vastly different scales, from lightweight edge deployments to large-scale cloud systems. The LLaMA series models developed by Meta are offered in various sizes from 7 billion to 70 billion parameters. However, this massive parameter size comes with significant challenges such as storage load, inference latency, and training scalability, limiting applicability, especially on resource-constrained devices (edge devices, mobile platforms, consumer-grade GPUs). As a result of various techniques used for model compression, model sizes are reduced and all the capabilities of the model are accessed by using less storage space. Although traditional methods of post-training quantization and structured/unstructured pruning have provided certain gains, accuracy losses are inevitable, especially in medium-sized models at the scale of 1-10 billion parameters [5]. These models represent challenging scales for deployment at the edge. In cloud-based model services, optimized models both use bandwidth efficiently and increase system performance by reducing processing time. Model optimization also contributes to energy efficiency, a crucial consideration in addressing the environmental and operational challenges of the 21st century. In addition to the technical performance of the compaction, the model is also critical to sustainability goals. When considered within the scope of sustainability goals, operating a smaller model means consuming less energy. While all these reasons reveal the importance of model optimization, significant gains are achieved both in terms of cost and system load.

In this study, firstly, LLaMA-based models published by Meta were obtained in .safetensors format and SparseGPT-based pruning method was applied on them. SparseGPT prunes unimportant parameters from a trained model without requiring full retraining [2]. SparseGPT pruning was applied on LlaMA-2 (7B, 13B) and LlaMA-3 (8B, 70B) models to analyze the effects on sparsity limits, perplexity, accuracy and inference rate. Sparsegpt, one of the model compression techniques, stands out with two features: it reduces the complexity of the model while not damaging the integrity of the architecture. It is an important criterion not to lose the accuracy of the model while pruning. This process, which is done layer by layer, is designed to preserve the overall accuracy of the model. This is the most important reason for the use of sparsegpt today, high performance is no longer sufficient, efficiency is also important.

In particular, the integration of lora and qlora provides the user with an option for task-specific optimization. Demonstrated retraining adaptability of billion-scale models on a single 24 GB GPU. These methods allow for fine-tuning and direct adaptation to the purpose without incurring performance and time overhead [6]. This capability makes the approach well-suited for scenarios requiring multiple task-specific model variants. This article presents a comprehensive methodology that systematically combines these techniques to achieve optimal trade-offs between model efficiency and performance, providing practitioners with a structured llm optimization approach that can be adapted to various model architectures and application areas. In this study, SparseGPT, LoRA and QloRA techniques are compared in terms of compression ratio, performance conservation and resource savings. While making the comparison, two main components were considered: LlaMA model series analysis and performance evaluation. The preparation for the first phase was to acquire LlaMA models of different sizes. Models with parameter sizes 7B, 13B and 70B were analyzed. The reason for choosing these models is to ensure that they are representative of real-world applications. In the second stage, the performances of SparseGPT, LoRA and QloRA techniques were measured, and comprehensive analyses were conducted with the results. The results are intended to be a guide in both academic and industrial applications. This paper presents a comprehensive methodology that systematically combines these techniques to achieve optimal trade-offs between model efficiency and performance, providing practitioners with a structured approach to LLM optimization that can be adapted to a variety of model architectures and application areas.

II. MATERIALS AND METHOD

This study is based on three-stage sparseGPT and LoRA/QLoRA optimizations for efficient compression of large language models. The method aims to achieve the optimal balance between model size, computational efficiency, and task performance.

A. General Pipeline Architecture and Model Acquisition

The proposed optimization pipeline consists of three consecutive stages: Model Acquisition and Calibration Data Preparation, followed by SparseGPT-based Layer-Level Pruning, and finally LoRA/QLoRA Fine-tuning Integration. In the experiments, LLaMA-2 (7B, 13B) and LLaMA-3 (8B, 70B) models were used. These models containing billions of parameters enable realistic evaluation of compression and parameter-efficient fine-tuning methods. The models were downloaded in .safetensors format. Two separate calibration datasets were prepared to support different stages of the optimization pipeline. The calibration data provided the foundation for Hessian computation in SparseGPT and offered representative samples for importance-based weight pruning.

B. WikiText-2 Dataset Preparation

Using the **prepare_dataset.py** script, the first 1000 sentences were extracted from WikiText-2 raw data and saved to the **calibration.txt** file. This data was used for Hessian estimation during pruning.

```
python
# prepare_dataset.py
dataset = load_dataset("wikitext", "wikitext-2-raw-v1", split="train")
with open("calibration.txt", "w", encoding="utf-8") as f:
    for i, example in enumerate(dataset):
        f.write(example["text"] + "\n")
        if i >= 999:  # Exactly 1000 examples
```

C. Alpaca Dataset Preparation

For task-oriented fine-tuning, the Alpaca dataset was converted to JSONL format using the **prepare_alpaca.py** script, resulting in the **alpaca_dataset.jsonl** file.

```
python
# prepare_alpaca.py
dataset = load_dataset("tatsu-lab/alpaca", split="train")
dataset.to_json("alpaca_dataset.jsonl", orient="records", lines=True)
```

This approach separated the calibration data used for pruning from the task data used for fine-tuning, thereby preventing interaction and contamination between the two stages.

D. Prunning with SparseGPT

Calibration datasets were prepared to support the pruning process. Models were pruned layer-by-layer using the SparseGPT algorithm (sparsegpt.py). Activation statistics were collected in each layer, and Hessian approximations were computed to zero out weights with low importance. Thus, the number of parameters was reduced while preserving the model's structure. The pruning operation was implemented in the llama.py file and tested on different datasets.

In the first step of the pruning process, Hessian approximation was utilized. Input activations were collected with the add_batch function, and a block-based Hessian approximation was constructed (1):

$$H \leftarrow H + XX\text{^}T \tag{1}$$

Subsequently, in the fasterprune function, weights with the smallest values scaled by the Hessian were identified and zeroed out.

E. Fine-tuning with LoRA and QLoRA

After pruning, LoRA adapters were added to recover the lost accuracy. The base model was reduced to 4-bit precision (quantized), while LoRA adapters were kept at high precision. LoRA was implemented by adding decomposition matrices transformer layers. The mathematical foundation can be summarized as shown in equation (2):

$$h = W_0x + \Delta Wx = W_0x + (BA)x \tag{2}$$

F. Quantizer

The Quantizer module developed in the quant.py file was used to reduce inference costs. Scale and zero-point parameters were learned according to weight distributions. Thus, memory and computational costs were significantly reduced while accuracy loss was kept to a minimum level.

The quantization process is defined as shown in equation (3):

$$Q = \text{round}(x/\text{scale}) + \text{zero} \tag{3}$$

III. RESULTS

This section presents the empirical evaluation of the three-stage optimization pipeline — SparseGPT pruning, LoRA fine-tuning, and quantization — applied to LLaMA-2 and LLaMA-3 models. Perplexity (PPL) is used as the primary metric [8].

A. SparseGPT Pruning

SparseGPT pruning was applied at 50% and 60% sparsity. As shown in Table I, pruning increased perplexity slightly across all models. At 50% sparsity, the average PPL increase remained under 3%, while 60% sparsity introduced a larger but still moderate rise.

B. LoRA Fine-Tuning

Introducing LoRA adapters reduced the PPL degradation caused by pruning. LoRA recovered approximately 70–75% of the lost accuracy across models.

C. Quantization

Quantization was applied after pruning and LoRA fine-tuning. 8-bit quantization introduced negligible degradation (<0.5% PPL), while 4-bit remained modest (<2%). The 50% Sparse + LoRA + 4-bit setup achieved over 75% memory reduction with minimal accuracy loss.

D. Overall Compression Efficiency

As shown in Table I, combining pruning, LoRA, and quantization provided substantial efficiency gains. The 50% Sparse + LoRA + 4-bit configuration achieved ~75% size reduction with a PPL increase of only 0.5, while 60% sparsity achieved ~80% reduction with 0.7 PPL increase. Inference throughput improved up to 3.4× compared to baseline.

Table I. Overall compression efficiency results

Technique	Size Reduction	PPL Increase	Speed-up
50% SparseGPT	~45%	+0.3	1.8×
60% SparseGPT	~55%	+0.6	2.1×
50% Sparse + LoRA	~45%	+0.15	1.8×
60% Sparse + LoRA	~55%	+0.3	2.1×
50% Sparse + LoRA + 4-bit	~75%	+0.5	3.2×
60% Sparse + LoRA + 4-bit	~80%	+0.7	3.4×

IV. DISCUSSION

The results show that combining SparseGPT pruning with LoRA/QLoRA fine-tuning and quantization works well for optimizing large models. When we applied SparseGPT pruning by itself, we saw significant parameter reduction without much accuracy loss. This matches what other researchers have found with unstructured pruning methods [9]. Adding LoRA adapters after pruning helped us get back most of the lost accuracy while keeping the efficiency gains [6]. This aligns with previous work on parameter efficient fine-tuning.

Adding quantization made the pipeline even better. We could reduce memory usage quite aggressively while only losing a small amount of accuracy in most cases. This fits with recent QLoRA research showing that low-rank adaptation works well with quantized models for real-world use [3]. What's particularly encouraging is that our three-stage approach worked across different model families - both LLaMA-2 and LLaMA-3 - and scaled from 7B all the way up to 70B parameters [7]. This suggests the method is robust and generalizable.

Looking at the bigger picture, this combination of techniques does more than just solve memory and computation problems. It also makes large AI systems more sustainable. Smaller models that run faster use less energy and cost less to operate. This matters for the environment and for making AI more economically viable. Our approach opens up possibilities for running powerful models on mobile devices, embedded systems, and edge computing platforms where you can't deploy full-sized models. This could make advanced AI accessible in many new contexts where it wasn't practical before.

V. CONCLUSION

Throughout this study, a systematic optimization was applied to large language models (LLMs) by continuing the SparseGPT-based pruning process with LoRA and QLoRA fine-tuning methods. The starting point of the study is that large language models such as Llama-2 and Llama-3 are not memory-efficient, making their use challenging in constrained hardware environments. The study began with dataset preparation, followed by layer-wise pruning with SparseGPT. Through this approach, 50-60% sparsification was achieved without the need to retrain the model. Subsequently, LoRA and QLoRA adapters were utilized to compensate for potential accuracy losses. Experimental results demonstrate that the conducted study reduced the model size by approximately half and decreased memory consumption by 40-55%. The results reveal that the integration of pruning, adapter-based fine-tuning, and quantization techniques within a single methodological framework offers significant benefits both in terms of practical utility and research perspectives. In conclusion, the integration of SparseGPT pruning with LoRA/QLoRA fine-tuning presents a powerful approach for more efficient, scalable, and sustainable deployment of large-scale models. This methodology constitutes an appropriate solution for both academic and industrial applications, particularly in scenarios with limited computational and memory resources.

ACKNOWLEDGMENT

The authors would like to acknowledge the open source resources that provided the underlying frameworks and datasets used by these developers. We also acknowledge the Hugging Face platform for providing model access in this work, Meta AI for publishing the LLaMA model family, and the research teams who made SparseGPT, LoRA, and QLoRA publicly available. Their contributions significantly enabled the reproducibility and extensibility of this work.

REFERENCES

[1] IDC, *Worldwide Global DataSphere Forecast, 2021–2025*. IDC, 2021.
[2] E. Frantar and D. Alistarh, "SparseGPT: Massive language models can be accurately pruned in one-shot," *arXiv preprint arXiv:2301.00774*, 2023. [Online]. Available: https://arxiv.org/abs/2301.00774
[3] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, "QLoRA: Efficient finetuning of quantized LLMs," *arXiv preprint arXiv:2305.14314*, 2023. [Online]. Available: https://arxiv.org/abs/2305.14314
[4] Meta AI, "LLaMA: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023. [Online]. Available: https://arxiv.org/abs/2302.13971
[5] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," in *Proc. Int. Conf. Learn. Representations (ICLR)*, 2016.
[6] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "LoRA: Low-rank adaptation of large language models," in *Proc. Int. Conf. Learn. Representations (ICLR)*, 2021.
[7] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, et al., "LLaMA 2: Open foundation and fine-tuned chat models," *arXiv preprint arXiv:2307.09288*, 2023. [Online]. Available: https://arxiv.org/abs/2307.09288
[8] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, et al., "Language models are few-shot learners," *arXiv preprint arXiv:2005.14165*, 2020. [Online]. Available: https://arxiv.org/abs/2005.14165
[9] E. Frantar, S. P. Singh, and D. Alistarh, "Optimal brain compression: A framework for accurate post-training quantization and pruning," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2022.