AI

MDPI

Article

Automated Pruning Framework for Large Language Models Using Combinatorial Optimization

Patcharapol Ratsapa [1], Kundjanasith Thonglek [1], Chantana Chantrapornchai [1,*] and Kohei Ichikawa [2]

[1] Department of Computer Engineering, Faculty of Engineering, Kasetsart University, Bangkok 10900, Thailand; patcharapol.ra@ku.th (P.R.); kundjanasith.t@ku.th (K.T.)
[2] Division of Information Science, Department of Science and Technology, Nara Institute of Science and Technology, Nara 630-0192, Japan; ichikawa@is.naist.jp
* Correspondence: fengcnc@ku.ac.th

Abstract: Currently, large language models (LLMs) have been utilized in many aspects of natural language processing. However, due to their significant size and high computational demands, large computational resources are required for deployment. In this research, we focus on the automated approach for size reduction of such a model. We propose the framework to perform the automated pruning based on combinatorial optimization. Two techniques were particularly studied, i.e., particle swarm optimization (PSO) and whale optimization algorithm (WOA). The model pruning problem was modeled as a combinatorial optimization task whose the goal is to minimize model size while maintaining model accuracy. The framework systematically explores the search space to identify the most optimal pruning configurations, removing redundant or non-contributory parameters. The two optimizations, PSO and WOA, were evaluated for their ability to efficiently navigate the search space. As a result, with PSO, the proposed framework can reduce the model size of Llama-3.1-70B by 13.44% while keeping the loss of model accuracy at 19.25%; with WOA, the model size reduction is 12.07% with 22.81% loss of model accuracy. Since accuracy degradation may occur during pruning process, the framework integrates the post-process to recover the model accuracy. After this process, the pruned model loss can reduce to 12.72% and 14.83% using PSO and WOA, respectively.

Keywords: large language models; model pruning; particle swarm optimization; resource-constrained devices; whale optimization algorithm

Academic Editor: Isidoros Perikos

Received: 30 March 2025
Revised: 28 April 2025
Accepted: 30 April 2025
Published: 5 May 2025

Citation: Ratsapa, P.; Thonglek, K.; Chantrapornchai, C.; Ichikawa, K. Automated Pruning Framework for Large Language Models Using Combinatorial Optimization. *AI* **2025**, *6*, 96. https://doi.org/10.3390/ai6050096

1. Introduction

Large language models (LLMs), such as OpenAI's GPT [1], Google's Gemini [2], Meta's Llama [3], and similar architectures, have fundamentally reshaped numerous fields over the past decade, pushing the boundaries of what artificial intelligence can achieve in natural language understanding, generation, and reasoning. These models, which leverage vast amounts of text data and sophisticated neural network architectures [4], are trained to understand, generate, and manipulate human language with remarkable fluency and coherence. The capabilities of LLM extend far beyond traditional natural language processing (NLP) tasks such as translation, sentiment analysis, and text summarization [5]. LLMs enable more complex applications that are beginning to pervade every aspect of modern society. The impact of LLMs also influences economic and social dynamics. Businesses use LLMs to gain a competitive edge, while educational institutions use them to create personalized learning experiences.

Driven by advances in computational power, data availability, and algorithmic innovation, the rapid development of LLMs has been unprecedented. Figure 1 illustrates the exponential growth in model parameters in recent years [6]. As models become larger, their ability to understand and generate human-like text improves, enabling them to perform a wide range of tasks, from answering questions and translating languages to composing creative content. One of the most remarkable examples of this trend is Meta's Llama, which was released with different variants, ranging from 7 billion to 405 billion parameters [7].

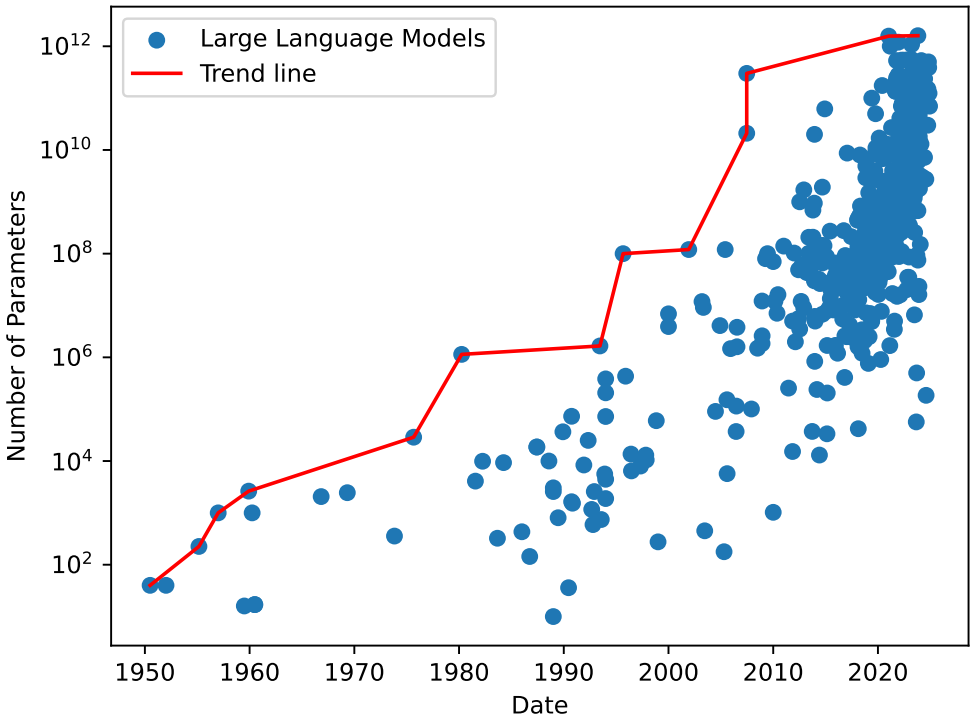


Figure 1. Parameter growth of large language models [6], based on data from [8].

Utilizing LLM on a large scale incurs hardware deployment expenses. For example, the impressive scale of Llama, which comes in variants spanning from 7 to 405 billion parameters, necessitates hardware such as GPUs with substantial VRAM (e.g., at least 140 GB for the 70 B variant in FP16 precision or around 35 GB with 4-bit quantization) or TPUs, along with considerable RAM and storage capacity to efficiently handle model weights and computations. This presents considerable challenges for its application on descent devices, including personal mobile devices or computers [9]. The larger the model, the more computational power, memory, and storage it requires. In particular, there are many use cases to deploy LLMs on personal devices that require data privacy and local model training [10].

Several approaches exist to minimize the size of the model. For example, knowledge distillation can transfer information from the teacher model into a smaller student model, although it often requires significant computational resources during the training process. Quantization can drastically compress the model size, but it can degrade the model accuracy. With the traditional pruning approach, the redundant parameters can be removed to effectively reduce the model and computational resource usages. However, identifying which parameters to be pruned is a challenge.

In particular, pruning is inherently a combinatorial problem. It involves selecting a subset of parameters from a potentially vast number of possibilities, where the goal is to optimize the trade-off between model size, accuracy, and execution time. The exponential search space of parameter subsets and the interdependencies between parameters make

the identification of the optimal configuration computationally complex. Thus, typical evolutionary algorithms, such as whale optimization algorithm (WOA), particle swarm optimization (PSO), ant colony optimization (ACO), genetic algorithm (GA), and others, are commonly applied as heuristics to selecting candidates to prune.

In this paper, we present the framework of automated layer pruning of model parameters that adopts evolutionary algorithms as heuristics to justify the value of parameters. One of the key challenges in pruning large-scale models is maintaining model accuracy while reducing computational costs. Traditional pruning techniques often rely on some user-defined heuristics, which may not generalize well across different architectures and datasets. We, instead, focus on evolutionary algorithms to iteratively optimize the pruning strategy, which selects and removes redundant parameters based on their contribution to the performance of the model. The results demonstrate a balanced trade-off between accuracy and model size by using PSO and WOA, enabling more effective model compression and making advanced AI capabilities more feasible on resource-constrained hardware. This advancement supports the deployment of LLMs in a variety of practical applications, such as real-time virtual assistants on mobile devices, personalized healthcare systems requiring on-device processing for privacy preservation, AI-powered educational tools that adapt to student feedback instantly, and offline legal document summarization where data confidentiality is critical. These scenarios demonstrate how efficient pruning can broaden the usability of LLMs across domains that demand both high performance and resource efficiency.

The structure of this manuscript is as follows. In Section 2, we provide an overview of existing techniques for implementing large language models (LLMs) and model compression. Section 3 outlines the framework and algorithms for reducing the size of LLMs while preserving accuracy and minimizing inference time. Section 4 presents experimental results, where we compare the effectiveness of PSO and WOA for model pruning heuristics. Finally, Section 6 offers concluding remarks and discusses future research directions.

2. Related Work

This section provides an overview of the existing concepts and techniques for implementing large language models. Model compression techniques are also discussed.

2.1. Large Language Models

Several large language models exist currently with several usages and applications.

Table 1 presents the comparison between each of the leading LLM performance metrics with their names highlighted in bold, suitability of the use case, architecture, and some of its variants. Each of these models has distinct strengths, making them suitable for different applications: GPT-4 has high accuracy and low perplexity, highlighting its performance in natural language generation tasks such as creative writing, chatbots, and coding assistance. The second row presents the Google AI, Gemini, which excels in strong reasoning and creative generations, gaining capability to suit in various domains. This makes the model useful in visual question answering, story telling, and image generation tasks. Claude prioritizes the safety and ethical usage of AI. Mistral advances its lower number of parameters and provides more proficiency in a strong regional context.

In the last row, Meta AI's Llama model is designed with research and academic applications in mind. It offers competitive performance while being open source, making it accessible for further customization and improvement by the research community. With a focus on publicly available data, Llama ensures greater transparency in training data and model behavior, making it a reliable choice for scientific studies and open AI devel-

opment. Llama-3.1-70B emerges as the ideal choice for this study, balancing performance, accessibility, and transparency in AI research [16].

Table 1. Comprehensive comparison of leading large language models (LLMs).

Model	Author	Performance	Use Case	Params	Variant
GPT	OpenAI [11]	High accuracy, low perplexity	Creative writing, chatbots, coding	1.8 T N/A	GPT-4o GPT-03-mini
Gemini	Google [12]	Strong reasoning	Creative Applications	>200 B	Gemini 1.5 pro
Claude	Anthropic [13]	Extended thinking, strong safety	Safety-critical applications	N/A N/A N/A	Claude 3.7 Sonnet Claude 3.5 Haiku Claude 3 OPUS
Mistral	Mistral AI [14]	Efficient, Low params	Real-time apps, programming	8 B 24 B	Ministral 8 B Mistral Saba
LlamA	Meta [15]	Competitive scores	Research, academia, open source	1 B, 3 B, 11 B, 90 B 70 B, 8 B, 405 B	Llama 3.2 Llama 3.1

Llama architecture incorporates several state-of-the-art techniques, such as multi-head self-attention and layer normalization, to achieve high performance [15]. With 70 billion parameters, Llama-3.1-70B is recognized as the second-largest model in the Llama3.1 series, allowing it to capture a broader range of linguistic nuances and complex patterns [17].

This study focuses on the automated pruning of large language models (LLMs) using combinatorial optimization, aiming to enhance model efficiency while preserving performance. By systematically selecting and removing redundant layers, the proposed framework seeks to reduce computational costs without significantly degrading the model's accuracy. This approach is particularly beneficial for deploying LLMs in resource-constrained environments, such as edge devices and mobile platforms, where efficiency is crucial.

2.2. Model Compression Techniques

Compressing neural network models is an essential strategy for reducing their size and computational complexity. However, it is important to ensure that these reductions do not come at the expense of model performance. Particularly, maintaining accuracy remains a difficult challenge. Hossain et al. identified seven key techniques for model compression, each offering different benefits and drawbacks: pruning, quantization, knowledge distillation, multiplication reduction, resource-efficient architectures, low-rank approximation, and network architecture search [18].

Pruning involves removing less important weights or neurons from the model, which reduces the overall parameter count and can lead to significant reductions in memory usage and computational overhead. However, pruning must be done cautiously, as it can lead to the accuracy degradation if critical weights are removed [19]. The key challenge with the pruning approach is the identification of the eliminated components [20].

Quantization, on the other hand, reduces the numerical precision of the model's weights and activations, for example, from 32-bit floating point to lower-bit representations (e.g., 8-bit integers). This can also drastically reduce storage requirements and speed up inference, especially on hardware that supports lower precision arithmetic [21]. However, quantization can also introduce quantization noise, which might degrade the model's performance, especially if the model is highly sensitive to small variations in weight values.

Knowledge distillation introduces the training of a smaller and efficient model called a student model to replicate the behavior of a larger, more accurate teacher model [22]. Despite its reduced size, the student model can achieve promising performance close to the teacher model. However, this technique may not work well when there is a significant

performance gap between the teacher and student models, as the student may struggle to generalize complex patterns learned by the teacher.

Other techniques include the multiplication reduction, through weight sharing or utilizing the tensor decomposition to reduce the total size of weight matrices [23]. This can enhance efficiency; however, it may require specialized hardware or network architecture transformation. Resource-efficient architectures employ lightweight layers and fewer parameters, making them task-dependent but effective in reducing computation. Low-rank approximation compresses models by approximating weight matrices with lower-rank versions, though it may introduce approximation error, which could affect model accuracy [24]. Network architecture search (NAS) automates the discovery of optimized architectures but is computationally intensive and typically does not consider total weight reductions [25].

Among these methods, the pruning approach offers a direct and intuitive way to reduce the model parameters. It consumes relatively low computational overhead compared to techniques like NAS or knowledge distillation [18]. Furthermore, pruning particularly reduces the number of parameters without majorly changing the network architecture. It also does not require the retraining process, which can consume computation resources. By carefully pruning a few parts of the network and fine-tuning the remaining components, it is possible to achieve significant model compression while retaining high accuracy.

2.3. Pruning Methods

Pruning methods vary according to criteria to select the removed weights or neurons. Table 2 summarizes several pruning techniques. Common selection approaches include the following.

Table 2. Comparison of pruning methods.

Pruning Method	Advantages	Disadvantages
Magnitude-based Pruning	Removes least important weights	Requires retraining to recover accuracy, may remove the important parameters
Structured Pruning	Optimizes hardware efficiency, improves inference speed	Requires retraining
Unstructured Pruning	Leads to sparse models, can greatly reduce parameter count	Requires specialized hardware for efficient execution
Iterative Pruning	Helps maintain accuracy by gradual pruning	Increases training time due to multiple pruning cycles
Global Pruning	More effective in removing unimportant weights across the model	Computationally expensive and difficult to tune
Local Pruning	Easier to implement with per-layer thresholds	Less optimal than global pruning for overall efficiency
Layer Pruning	Reduces model complexity significantly, improves execution speed	Requires retraining

Magnitude-based pruning removes the smallest weights, assuming that they contribute less to performance. Structured pruning removes entire neurons, channels, or layers, optimizing hardware efficiency but requiring retraining. Unstructured pruning removes individual weights, leading to sparse models that need specialized hardware.

Iterative pruning gradually removes parameters over multiple training cycles, helping maintain accuracy but increasing training time. Global pruning selects the least important

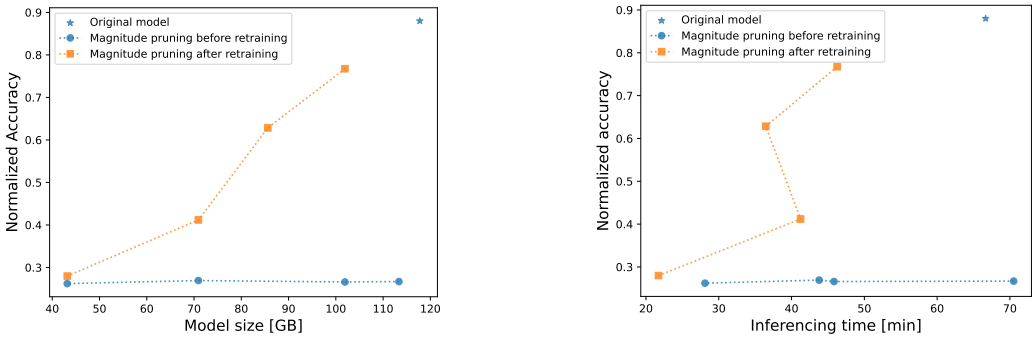
weights network-wide, while local pruning applies thresholds per layer. Global pruning is more effective, but computationally expensive.

From all of the pruning methods, layer pruning focuses on selectively removing entire layers or blocks of neurons within a neural network, rather than pruning individual parameters or neurons. This method offers a significant advantage in reducing the complexity of the model while maintaining the integrity of its structure. Pruning entire layers helps eliminate redundant computations and accelerates the model's execution without the need for complex parameter-level adjustments. However, it may limit the search space to entire layers rather than all individual parameters, restricting the model's ability to fine-tune its structure for optimal performance.

Many studies have proposed using heuristics to prune the layers. Zhang et al. utilized layer pruning for reducing network complexity [26]. A new method called layer pruning via structured residual (LPSR) was proposed. The pruning process involves estimating the impact of each block using Taylor expansion and batch normalization (BN) parameters, ensuring that only the least important blocks are removed.

Other research examples include [27], which introduces a way to select the pruned layer using a one-shot layer evaluation method using a proxy classifier through imprinting, resulting in a significant reduction in computational overhead. Tang et al. proposed a novel layer pruning method by exploring the stochastic re-initialization approach [28]. By giving the importance of each layer to the sensitivity in stochastic re-initialization (low accuracy drop), a great reduction in the number of parameters can be achieved.

Figure 2 shows the results of magnitude layer pruning, by summing the layer's weight and pruning the lowest magnitude layer. While this method has good potential in preserving accuracy, it has limited effectiveness without retraining. The removal of low-magnitude layers may disrupt important learned representations, leading to performance degradation [29]. Additionally, this approach does not always guarantee a reduction in inference time, as certain layers may contribute more significantly to computational efficiency. Even after retraining, the improvement in latency might be marginal if the pruned layers were not computationally expensive in the first place.



(**a**) Varying prune ratios (**b**) Varying inferencing time

Figure 2. Normalized accuracy for varying prune ratios (**a**) and inferencing time (**b**).

Several heuristic approaches, such as particle swarm optimization (PSO), whale optimization algorithm (WOA), and other combinatorial optimization algorithms, have been widely used in optimization problems. Liang et al. applied the multi-objective version of PSO to reduce its risk in portfolio optimization, demonstrating that multi-objective dynamic multi-swarm particle swarm optimization was a potential solution for large-scale asset allocation problems [30]. WOA shows its potential in [31], presenting a new PID control method based on the whale algorithm to optimize the BP neural network, increasing the response speed and stability.

There are similar studies that focus on the optimization algorithm of the pruning method. For example, Wong et al. applied the bee colony optimization (BCO) algorithm to solve the symmetric traveling salesman problem (TSP) [32]. This work extends the application of BCO by integrating frequent-closed-pattern-based pruning strategy (FCPBPS). The proposed FCPBPS enhances the efficiency of the BCO algorithm by leveraging frequent closed patterns mined using the BIDE algorithm, allowing non-promising solutions to be pruned before undergoing a local search. This reduces unnecessary computations while maintaining high-quality solutions.

PSO and WOA were chosen for their effectiveness and rapid convergence in handling high-dimensional combinatorial optimization problems. PSO provides strong global search capabilities, while WOA is effective in balancing exploration and exploitation, reducing the chances of premature convergence. Preliminary experiments also indicated that other metaheuristic algorithms, such as genetic algorithms (GA) or simulated annealing (SA), although frequently used in optimization, have significantly greater evaluation and search times for pruning optimization. GA often suffered from slow convergence due to the stochastic nature of crossover and mutation operations, which may not always efficiently refine candidate solutions. Similarly, SA, despite its theoretical ability to escape local minima, exhibited slower cooling schedules and a strong dependence on hyperparameter tuning, making it less practical for the high-dimensional layer pruning problem. Both GA and SA also demonstrated difficulties in consistently locating a globally optimal solution in the complex and large pruning search space [33].

Beyond their performance in this specific context, algorithms like PSO and WOA have become powerful tools in model compression and optimization due to their robust ability to navigate large, complex solution spaces [30,34]. Their success stems from their nature-inspired strategies, which provide adaptability across diverse problem domains and deliver consistently competitive results, particularly in combinatorial optimization challenges.

The particle swarm optimization (PSO) algorithm is inspired by the collective behavior of animals [35], such as birds or fish, that work together in swarms to find the optimal food source. Each individual in the swarm adjusts its position based on both its own experience and the experiences of its neighbors, which enables the swarm as a whole to converge toward the global optimum. PSO has gained widespread attention because of its simplicity, ease of implementation, and the impressive results it has shown across various domains, from function optimization to neural network training and model pruning. The core strength of PSO lies in its ability to balance exploration (searching the solution space) and exploitation (refining known solutions), which helps it efficiently navigate complex, high-dimensional spaces. This makes PSO particularly effective for problems with multiple local optima, where other algorithms might prematurely converge to suboptimal solutions.

WOA is inspired by the bubble-net feeding strategy of humpback whales [36], who create a spiral pattern in the water to trap fish. The WOA algorithm mimics the behavior, using a series of mathematical operations to simulate spiral movement and search for the best solution. What makes WOA particularly compelling is its ability to perform random exploration effectively, allowing one to avoid getting stuck in local optima, a common issue in many optimization algorithms. This characteristic allows WOA to maintain a robust search in complex solution spaces, making it a strong candidate for solving challenging combinatorial problems where traditional methods often fail or take longer to converge [37]. WOA's ability to blend exploration with exploitation in a way that prevents premature convergence makes it a powerful metaheuristic, particularly for high-dimensional or multi-modal problems.

Given the advantages of these two algorithms, we apply them in the context of pruning large language models (LLMs). In the previous section, we discussed various techniques for

compressing models and reducing their size while maintaining accuracy. Among the many methods available, automatic pruning using optimization algorithms like PSO and WOA offers a promising approach [38]. These algorithms can effectively search the large, high-dimensional space of possible pruned networks, identifying which layers or parameters are most suitable for pruning while minimizing the impact on model performance. Using the PSO and WOA algorithms, we aim to automate the pruning process, enabling us to find the optimal pruned layers that balance size reduction with accuracy retention. This approach not only improves model efficiency but also reduces the time and computational resources typically required for manual tuning.

The next section discusses how we apply these algorithms to prune LLMs and evaluate their effectiveness in finding the optimal pruned structure.

3. Materials and Methods

This section first outlines the proposed method for automatically pruning large language models (LLMs) using combinatorial optimization techniques. Figure 3 shows the overall process. The process begins with a careful analysis of the LLM architecture to identify which layers are suitable for pruning. Once we have determined the layers that can be pruned, the combinatorial optimization loop is an iterative process, where the goal is to find the optimal combination of layers to prune. The optimization process uses the size, accuracy, and inference time of the pruned model as objective values to guide the search for the best pruning configuration that balances the accuracy and inference efficiency of the model.

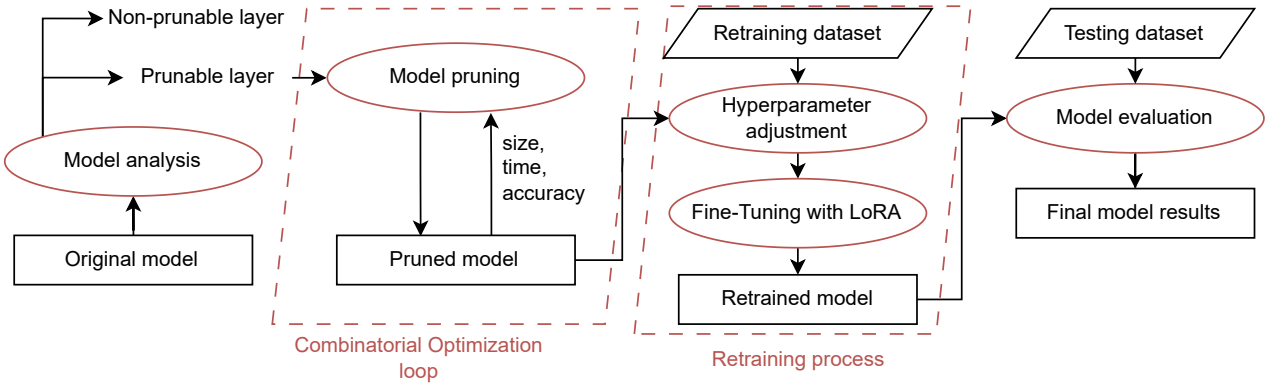


Figure 3. Overview of the proposed method.

After completion of the combinatorial optimization, the set of pruned layers is obtained. Then, they are used to retrain the pruned model. This retraining step is crucial to recover any potential accuracy degradation caused by pruning. Finally, the pruned and retrained model is evaluated to assess its accuracy. It is important to note that, for a fair comparison, the dataset used for retraining and evaluation differs from the one used for the original model training.

3.1. Model Analysis

In this paper, since Llama-3.1-70B serves as a case study for evaluating pruning strategies in large-scale language models, we describe more about the architecture of Llama-3.1 as follows. Llama-3.1-70B is a large language model developed by Meta, based on an auto-regressive architecture that leverages an optimized transformer design. With 70 billion parameters, Llama-3.1-70B represents a state-of-the-art advancement in transformer-based architectures.

Figure 4 illustrates the transformer-based architecture of Llama-3.1-70B, which consists of three main components: the embedding layer, transformer blocks, and a final normalization layer followed by a linear layer. The embedding layer is responsible for converting input tokens into dense vector representations that capture semantic information. The core of the model consists of 80 transformer blocks, which serve as the primary computational units.

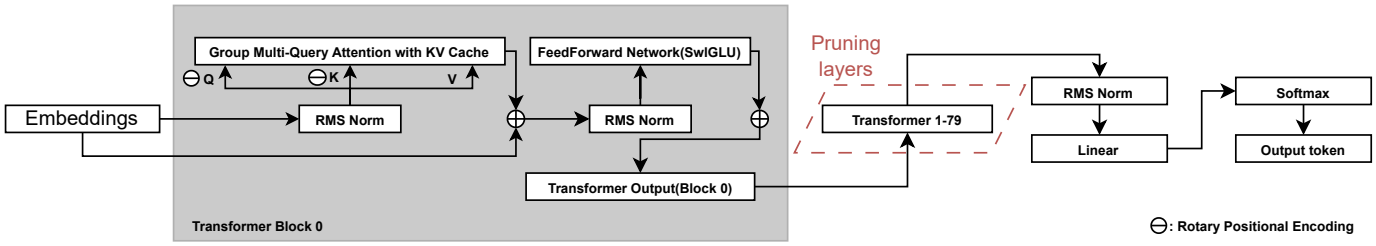


Figure 4. Llama-3.1-70B model structure.

These transformer blocks are stacked sequentially, each consisting of multi-head self-attention mechanisms and position-wise feedforward neural networks. The blocks are responsible for capturing complex dependencies and hierarchical patterns within the input data. After passing through the transformer blocks, the model applies a normalization layer to stabilize the the training, followed by a linear layer that produces the final output. These 80 transformer blocks directly correspond to the layers, with each block adding depth to the model and enhancing its capacity to model intricate language patterns.

There are three key components in Llama's layer:

1. Multi-Head Self-Attention Mechanism: Enables the model to capture a wide range of contextual relationships by projecting inputs into multiple attention subspaces;
2. Feedforward Neural Networks: Position-wise feedforward layers in each transformer block boost the model to learn complex representations;
3. Rotary Positional Embeddings (RoPE): Enables the model to process long sequences by including relative positional information directly into the attention.

3.2. Model Pruning

Pruning involves reducing the size of a model by removing a subset of layers, with the goal of minimizing any loss in accuracy. In the case of the above Llama model, all of these layers are potential candidates for pruning, although the number of layers that can be effectively pruned depends on task-specific requirements and performance constraints.

The pruning process requires identifying the optimal set of layers to retain or remove, which transforms the problem into a combinatorial optimization challenge. For example, if up to k layers are pruned from a total of $n = 80$ layers, the total number of possible pruning configurations is given by Equation (1).

$$TotalConfigurations = \sum_{i=0}^{k} \binom{80}{i} \tag{1}$$

When k is set to 40, the total number of possible pruning configurations grows exponentially, resulting in approximately $2^{79} \approx 6.04 \times 10^{23}$ possible combinations. This massive search space represents a significant computational challenge, as evaluating every possible configuration exhaustively would require enormous computational resources and time. As a result, performing an exhaustive evaluation of all options becomes practically infeasible.

3.3. Combinatorial Optimization Algorithms

Combinatorial optimization is a field of optimization that focuses on selecting the best solution from a finite set of possible solutions. In the case of model layer pruning, our goal is to identify which layers of a neural network should be removed in order to reduce its size and computational complexity while maintaining as much of its predictive accuracy as possible. Given the large number of potential pruning configurations, the problem quickly becomes a combinatorial one, where the number of possible solutions grows exponentially with the number of layers in the model.

Metaheuristics are global search algorithms that guide the search process towards the best possible solution without needing to evaluate every potential configuration. These methods, inspired by natural phenomena or processes, are particularly useful in problems like model pruning where the solution space is too vast to explore exhaustively. Among the many metaheuristic algorithms available, particle swarm optimization (PSO) and whale optimization algorithm (WOA) have proven to be especially effective for combinatorial optimization tasks like pruning. In our example, we use PSO and WOA as case studies to evaluate the effectiveness of our proposed framework, analyzing their performance in optimizing model pruning.

In our framework, we apply WOA and PSO algorithms to search for the optimal set of pruned layers while maintaining a balance between accuracy, model size, and inference efficiency. The balance between aggressive pruning and accuracy retention is inherently handled by the optimization behavior of each algorithm. Each algorithm evaluates candidate pruning strategies based on a fitness function that penalizes severe accuracy loss while encouraging model sparsity. Therefore, the ability to maintain this balance is not fixed by the framework itself, but rather emerges from the search dynamics and selection criteria of the optimization algorithm being used. We define the search space as a subset selection problem where we configure the number of pruned layers as input; for the Llama-3.1-70B model, there are 80 available layers. Each candidate solution is represented as an ordered list of pruned layer indices:

$$S = \{l_1, l_2, \ldots, l_k\}, \quad \text{where } l_i \in \{1, 2, \ldots, 80\}, \quad k = \text{number of pruned layers.} \tag{2}$$

For example, when pruning 10 layers, the variable S will be

$$S = \{3, 7, 12, 15, 19, 24, 28, 30, 35, 40\} \tag{3}$$

Layers 3, 7, 12, ... 40 will be pruned while others are retained.

3.4. Model Retraining

After the selected layers are pruned, the new structure of the model is retrained again to fine-tune parameters in order to improve the accuracy. The fine-tuned neural network model is achieved by repeating the training step at least 4000 iterations. This can help the pruned LLM recover the accuracy after the impact of pruning. In this work, the retraining process follows two stages:

1. Hyperparameter Adjustment: To improve convergence and stability after pruning, key hyperparameters, such as learning rate, batch size, and number of epochs, are carefully tuned.
2. Fine-Tuning with LoRA: The pruned model is fine-tuned using a general-domain dataset, enhanced by incorporating low-rank adaptation (LoRA) [39]. Unlike conventional fine-tuning that updates all model parameters, LoRA introduces lightweight and efficient parameter updates by injecting trainable low-rank matrices into the model, allowing effective adaptation with a smaller number of parameters. This

approach requires less computational overhead and prevents overfitting by focusing on a smaller set of parameters. This method helps the pruned model to adjust its remaining parameters effectively while maintaining versatility and generalization across a wide range of downstream tasks.

3.5. Model Evaluation

To facilitate model evaluation, our framework leverages the Language Model Evaluation Harness [40], an open-source toolkit that provides a standardized and flexible benchmarking setup for LLMs. This framework is compatible with models from the Hugging Face ecosystem, supporting popular architectures such as GPT, Gemma 2, and BERT. Therefore, as long as the evaluation harness supports the target model and the model is based on a Transformer architecture, our framework generalizes well without major modifications, making it versatile across a wide range of LLM architectures.

The derived model is evaluated against the unknown dataset. In our experiments, we tested against the HellaSwag dataset [41], a widely recognized benchmark for commonsense reasoning tasks. The evaluation process for the pruned models was multifaceted, measuring several critical performance metrics, including accuracy, model size, and inference time.

In addition to precision, the size of the pruned models was measured. Inference time was recorded to evaluate the computational efficiency of the pruned models during the evaluation phase.

4. Experimental Results

The proposed method was evaluated against the following aspects: model size, accuracy, and inference time. First, we describe the experimental setup used in this study, including the model and hardware specifications. Next, we present the results of applying different combinatorial optimization techniques, namely PSO and WOA. Finally, we compare the performance of the pruned models with the original model, highlighting the outcomes of both PSO and WOA-based pruning.

4.1. Experimental Setup

Table 3 provides the hardware specifications used in our experiments. We selected Llama-3.1-70B [42], a widely adopted large language model (LLM) with 70 billion parameters, as the base model for pruning and optimization.

Table 3. Hardware specification.

Hardware	Specification
CPU	Xeon Gold 62.30R 26 core ×2
Main Memory	256 GiB
GPU	NVIDIA A100 ×2
GPU Memory	40 GiB ×2

The optimization process was carried out using the Mealpy library [43], which facilitated the implementation of PSO and WOA in a multi-objective optimization framework. Specifically, we employed the weighted-sum method to combine multiple objectives, such as minimizing size and inference time while maintaining accuracy, allowing for an efficient exploration of the trade-offs between these competing factors.

4.2. PSO Results

We observed how PSO optimization progressively balances the trade-off between maintaining model accuracy and reducing both model size and inference time. By tracking these metrics over each step, we aim to understand how effectively the algorithm can optimize these competing objectives simultaneously.

Figure 5 illustrates the normalized accuracy achieved during each of 10 steps of PSO optimization, with the star-shape signifying the lowest accuracy for each number of pruned layers. Normalized accuracy is the accuracy adjusted using byte-length normalization, a method where the log probability of a continuation is divided by the number of bytes it occupies. This ensures the consistency of the scoring across different tokenization schemes, preventing models from being unfairly penalized for generating longer or shorter outputs.

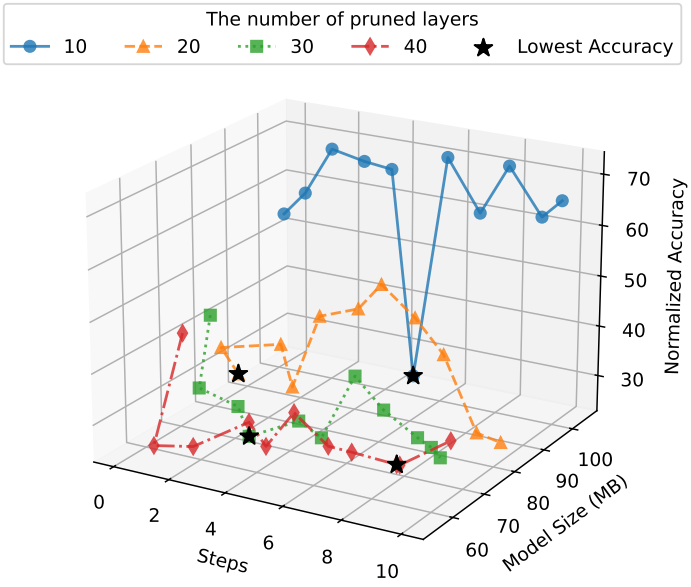


Figure 5. Model's size and normalized accuracy during PSO optimization.

For 10 pruned layers, the model still maintains most of its performance, showing a relatively high normalized accuracy across all optimization steps. Although occasional drops occur, such as around steps 5 and 8, the others step's accuracy remains stable.

For 20 pruned layers, the accuracy exhibits greater fluctuations compared to the 10-layer case. A peak accuracy is observed around step 6, while a significant accuracy drop occurs in later steps, particularly at step 9. While this configuration still retains acceptable accuracy, the increased pruning introduces greater sensitivity to optimization variations.

As for 30 pruned layers, the normalized accuracy becomes more unstable compared to fewer pruned layers. The trend follows a similar pattern as 20 pruned layers. Thus, it implies that, as more layers are removed, the model struggles to maintain its predictive capability, leading to an increased likelihood of accuracy loss.

In the case of 40 pruned layers, the model exhibits the lowest and most unstable accuracy values among all configurations. Excessive layer pruning results in a drastic reduction in the model's ability to generalize effectively.

We observed that seven specific pruned layers consistently appear in the steps with the lowest accuracy, with each layer being selected three times across different optimization steps.

This suggests that these layers play significant roles in maintaining the model's overall accuracy. Specifically, in the fifth step, the case where ten layers were pruned, three of these seven layers were removed, resulting in a sharp decline in accuracy, as shown with the

black star. Conversely, the 52nd and 37th layers were the most frequently pruned in the steps with the highest accuracy, indicating their potential importance in preserving model performance throughout the optimization. Note that these layers, 52nd and 37th, both have the number of parameters equal to 855,654,400 parameters.

Figure 5 depicts the change in model size as a result of the PSO optimization process. As expected, an increase in the number of pruned layers correlates with a reduction in the overall model size. A key observation is that the 38th layer was selected for pruning in every configuration, consistently resulting in the smallest model size for each corresponding pruning iteration. This pattern implies the 38th layer's significant contribution to reducing model size, suggesting that it may have a more direct impact on the efficiency of the pruning process compared to other layers.

The inference time during PSO optimization is shown in Figure 6. While the general trend indicates that pruning more layers tends to reduce inference time, there are instances where models with fewer pruned layers exhibit longer inference times than those with more layers pruned. Further investigation revealed that the 38th and 46th layers were the most frequently pruned in the optimization steps associated with higher inference times.

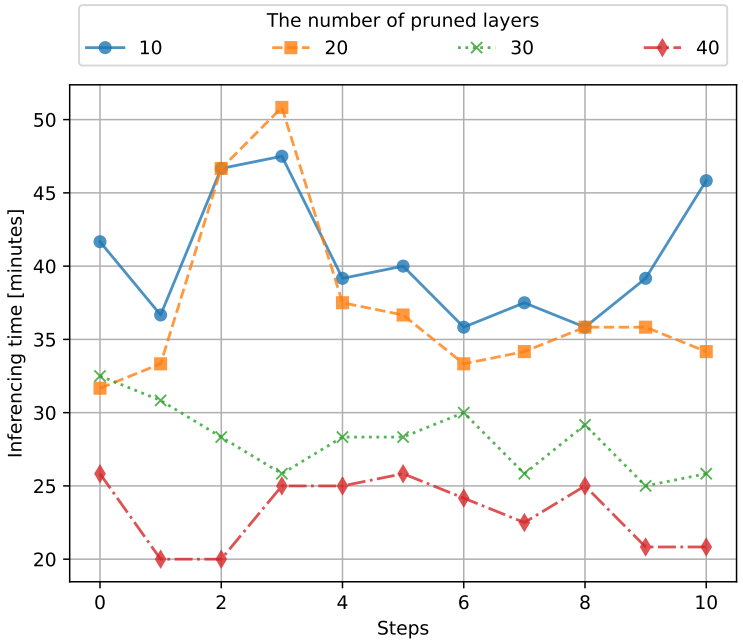
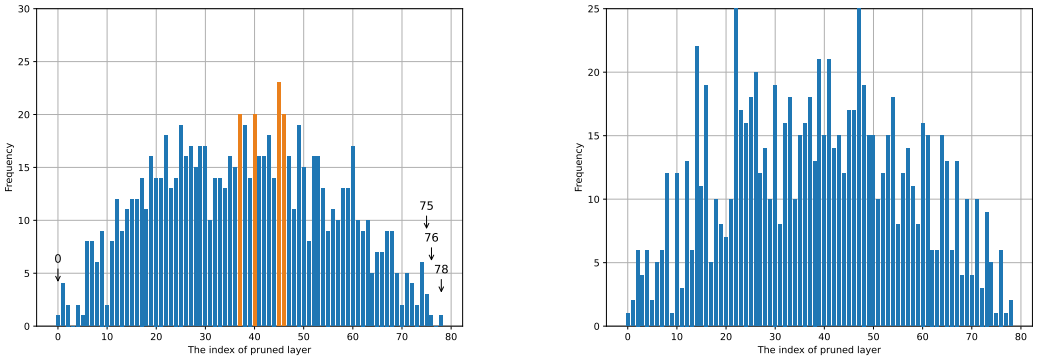


Figure 6. Model's inferencing time during PSO optimization.

Interestingly, the 38th layer, which plays a significant role in minimizing model size, presents a potential contradiction: while its pruning reduces the overall size of the model, it does not consistently result in a reduction in inference time. This observation prompted us to further investigate the layers most and least frequently pruned during the PSO optimization process.

The total number of times each layer was selected for pruning is shown in Figure 7a. Among the layers, the 37th, 40th, 45th, and 46th layers with the orange highlight in the figure were the most frequently pruned, with each being selected up to 20 times during the optimization process. Conversely, the 0th, 75th, 76th, and 78th layers had the lowest frequency of selection, being pruned only once, as indicated by the arrow in the figure. These findings suggest that layers with fewer selections, particularly those situated in the middle part of the model, were less significant. In contrast, layers that were more frequently pruned, especially those located near the input and output part of the model, seem to be significant in determining the model accuracy. The most important components of the

model are concentrated at its boundaries, while the central layers may be more dispensable in terms of pruning.



(**a**) Frequency of pruned layers with PSO (**b**) Frequency of pruned layers with WOA

Figure 7. Frequency of pruned layers with PSO (**a**) and WOA (**b**).

4.3. Results for Pruning with WOA

The results of the whale optimization algorithm (WOA) were similar to the PSO's. The WOA was used to explore the trade-offs between these objectives, with the goal of minimizing model size and inference time while maintaining accuracy. The results from WOA are analyzed and compared to those obtained from PSO to assess the relative effectiveness of both algorithms in pruning large-scale models like Llama-3.1 70B.

In terms of normalized accuracy, as shown in Figure 8, five specific pruned layers were selected across the steps with the lowest accuracy, each appearing three times. Notably, the 25th layer overlapped with the PSO results, further indicating its significant role in maintaining model accuracy. In contrast, the 14th, 16th, 41st, 56th, and 60th layers were the most frequently pruned in the steps with the highest accuracy, suggesting that these layers contribute substantially to the model's overall performance. This pattern of accuracy stabilization across optimization steps indicates that WOA, like PSO, is effective in preserving model accuracy while pruning.

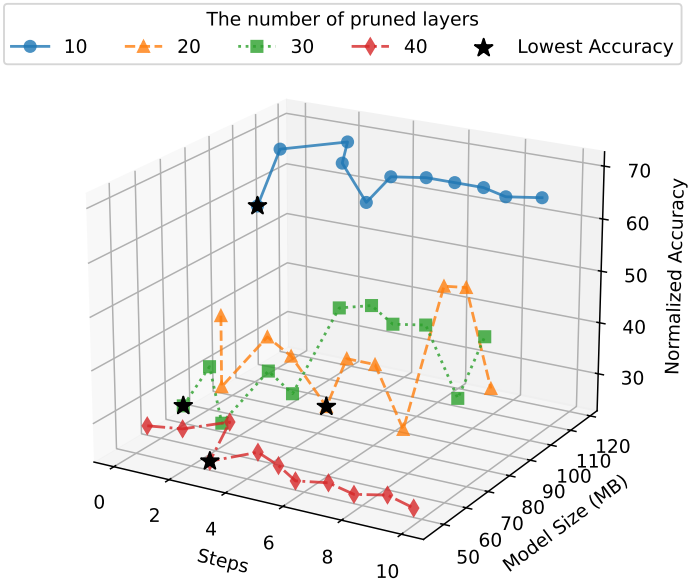


Figure 8. Model's size and normalized accuracy during WOA optimization.

Regarding model size, Figure 8 demonstrates that the 22nd layer was the most consistently pruned layer, leading to the smallest model size in each pruning step. This suggests that the 22nd layer plays a pivotal role in size reduction across different configurations. Interestingly, the 38th layer also had a significant impact on reducing model size, appearing consistently in the lowest size models, similar to the PSO results. When comparing layer frequency between PSO and WOA, the 38th layer stands out as particularly influential in size reduction, reinforcing its importance across both optimization methods. Figure 9 reveals a similar trend to that of PSO: models with 40 pruned layers exhibit the lowest inference time, while those with fewer pruned layers occasionally show longer inference times. Upon further analysis, we found that the 24th and 33rd layers were most frequently selected in the steps with the longest inference times.

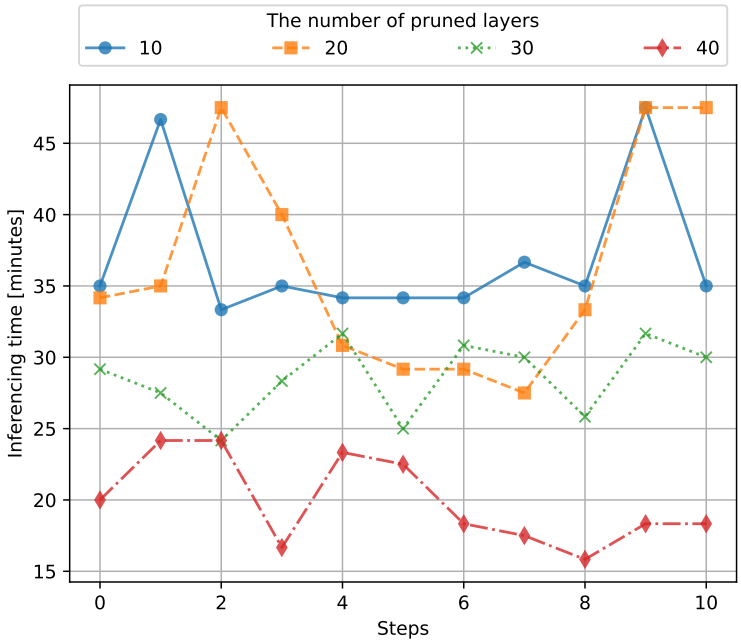


Figure 9. Model's inferencing time during WOA optimization.

However, the 22nd, 32nd, and 44th layers were predominantly selected in the steps with shorter inference times. This indicates that, while the pruning of certain layers can reduce inference time, the relationship between layer selection and inference time is complex, with some layers influencing time performance in unexpected ways. When comparing the results of PSO and WOA, both optimization algorithms show that pruning 10 layers generally maintains the highest and most stable accuracy, while pruning 40 layers leads to the lowest accuracy across all steps. However, the accuracy fluctuations are more pronounced in PSO, particularly for the 20- and 30-layer pruning configurations, while WOA demonstrates smoother variations in accuracy, indicating its ability to maintain more consistent performance.

For model size, both algorithms display a consistent trend, with WOA showing slightly more fluctuation in size for the 20- and 30-layer pruning configurations than PSO. Similarly, for inference time, both algorithms exhibit a similar pattern, with 40 pruned layers resulting in the lowest inference time. However, WOA offers better stability in inference time, showing fewer fluctuations than PSO. As a result, WOA appears to provide smoother performance across several metrics, including accuracy and inference time, compared to PSO, which tends to exhibit more pronounced fluctuations. While PSO's broader search capabilities allow it to escape local optima more effectively, WOA's smoother optimization

process suggests it may be better suited for scenarios where stability in accuracy, size, and efficiency is critical.

Figure 7b illustrates the total number of selected pruned layers, showing that the 14th, 22nd, 26th, 39th, 41st, and 47th layers were the most frequently pruned, each appearing more than 20 times. WOA demonstrates sharper peaks compared to PSO. On the other hand, the least frequently selected layers, appearing fewer than five times, were the 0th, 1st, 3rd, 5th, 11th, 68th, 70th, 72nd, 75th, 77th, and 78th layers, all located at the edges of the model, similar to PSO.

4.4. Comparison Between PSO and WOA

Focusing on the layers pruned by both algorithms, the most commonly overlapped layers were the 22nd, 25th, and 37th, each appearing 18 times. This suggests that both PSO and WOA consistently identify these layers as less critical to model performance, making them frequent candidates for pruning. In contrast, the 0th, 5th, 9th, 75th, 76th, and 78th layers, which were selected less frequently by both algorithms, may be deemed essential due to their potential contribution to accuracy or structural integrity.

Comparing Figure 7a,b, both algorithms tend to focus on the middle layers (21–59), likely because these layers contribute less to overall model performance. However, there are notable differences between the two methods. PSO exhibits a more evenly distributed frequency across the mid-range layers, with a peak around the 40th layer, while WOA shows sharper peaks, particularly around the 25th and 50th layers, indicating a more focused pruning strategy. Additionally, both algorithms prune edge layers (0–20 and 60–80) less frequently, suggesting these layers are more critical to maintaining accuracy.

Table 4 presents a comparison of the pruned models obtained using PSO and WOA after retraining. The row indicates the models obtained by either PSO or WOA with the given number of layers pruned. The search time, resulting model size, inference time, and normalized accuracy for each resulting model are shown. The results indicate that each algorithm has its strengths depending on the evaluation criteria.

Table 4. Comparison of pruned model using PSO and WOA after retrain.

Models	Avg Search Time (Min)	Red. Model Size (GB)	Infer. Time (Min)	Normalized acc.
PSO(10)	40.5303	101.9299	50.6833	0.7680
WOA(10)	36.9697	103.5619	49.4834	0.7495
PSO(20)	37.2727	74.1854	32.9	0.6274
WOA(20)	36.5151	74.1854	37.0667	0.2676
PSO(30)	28.1818	51.3369	24.5166	0.4239
WOA(30)	28.5606	51.3369	22.5333	0.4888
PSO(40)	23.1818	20.2383	13.7167	0.2676
WOA(40)	19.9242	20.2383	20.0	0.2637

In term of search time, WOA mostly takes less searching time. For larger number of pruning layer i.e., in 40 layers WOA only takes 19.92 min compared to PSO that use 23.18 Min, WOA may be more efficient in traversing the search space and identifying redundant layers. However, this case leads to very low accuracy.

Regarding model size reduction, both PSO and WOA achieve similar levels of compression across different pruning configurations. However PSO tends to achieve slightly more reduction in lower pruning layer such as 10 layer, PSO reveals its reduced size to 101.93 GB versus WOA which equals to 103.56 GB.

For inference time, the results show inconsistency depending on the number of pruned layers. For example, at 20 pruned layers, WOA has a longer inference time than PSO,

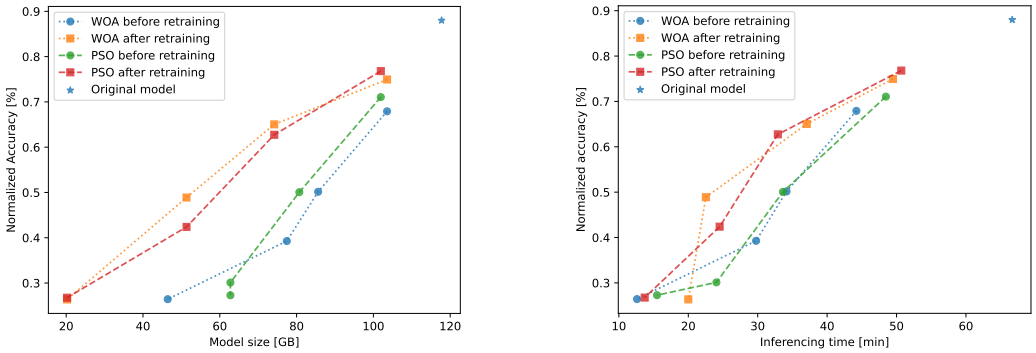
whereas at 30 layers, WOA is faster. This fluctuation suggests that the effectiveness of each algorithm in reducing inference time depends on the specific layers pruned rather than a consistent trend favoring one method over the other.

Consequently, accuracy differences become apparent when 30 layers are pruned. At this point, the WOA algorithm achieves an accuracy improvement of 15.31% over PSO. However, when pruning 20 layers, the accuracy gap narrows to 3.67%. For models with 10 pruned layers, PSO outperforms WOA by 2.47% in accuracy. These results suggest that the WOA is more effective for models smaller than 75 GB, while PSO delivers superior performance for larger models.

4.5. Compared to Original Model

After apply postprocessing, we measured the model size and accuracy again. The relationship between model size and accuracy is illustrated in Figure 10a, which compares the retrained model to its original model. WOA and PSO after the retraining step ensures that the reduced model maintains its performance while significantly decreasing its storage and computational requirements. As shown in the graph, the retraining process effectively reduces the model size. This reduction is achieved by integrating the LoRA weights directly into the model. Merging of LoRA weights contributes to a more compact model, while still retaining its performance characteristics. Both algorithms exhibit a similar rate of reduction, which is inversely proportional to the number of pruned layers.

Figure 10b shows the relationship between inference time and accuracy. While inference time appears to correlate directly with accuracy, the retraining process mitigates this trade-off. Specifically, the accuracy of the retrained model increases, while the inference time remains largely unchanged compared to the original model.



(**a**) Varying pruning ratios (**b**) Varying inferencing time

Figure 10. Normalized accuracy for varying pruning ratios (**a**) and inferencing time (**b**), comparing algorithms with and without retraining.

5. Discussion

The current framework provides a solid foundation for model pruning and optimization but would require key modifications to support dynamic, on-device pruning. While it already includes layer selection via combinatorial optimization, evaluation, pruning, and retraining capabilities, adapting it for continuous, resource-aware evolution would necessitate lightweight optimization modules for real-time decision-making. By integrating real-time monitoring of device resources and user-specific usage patterns, the framework could enable continuous pruning based on resource availability, usage trends, and performance requirements. Furthermore, modifying the existing pipeline to support incremental updates and runtime pruning decisions would allow models to evolve dynamically based on actual user behavior and device constraints.

6. Conclusions

We proposed an automated framework to reduce the size of LLM models that relies on the pruning approach. After the pruning, the derived model was applied postprocessing for dataset retraining, hyperparameter adjustment, and LORA fine tuning to improve the model accuracy. The framework was tested against two combinatorial methods, namely PSO and WOA,with the Llama-3.1 70B model. The results show that PSO achieves a 13.44% reduction in model size, with an associated accuracy loss of 12.72%. In comparison, WOA reduces the model size by approximately 12.07%, but with a higher accuracy loss of 14.83%.

Comparing the two combinatorial methods, we observed that WOA provides more balanced solutions than PSO, particularly for models smaller than 75 GB, where it delivers better accuracy. However, for larger models, PSO demonstrates an ability to maintain or improve accuracy more effectively than WOA. This suggests that WOA is more efficient for pruning smaller models, while PSO performs better with larger ones.

Further, the pruning results imply that the middle layers of the model, specifically those in the range of the 20th to 50th, are less important than the layers (0–20 and 60–80) since pruning them can lead to a decrement in the model performance. The framework allows us to understand that model layers' importance while comparing pruning strategy.

Future work will explore other pruning techniques on other LLMs and apply additional combinatorial optimizations. We also plan to investigate the integration of combinatorial optimization with other model compression techniques and assess additional factors such as power consumption and computational efficiency, particularly for real-world applications on resource-constrained devices.

Author Contributions: Conceptualization, P.R. and K.T.; methodology, P.R. and K.T.; software, P.R.; validation, P.R., K.T., C.C. and K.I.; formal analysis, P.R.; investigation, P.R.; resources, K.I.; data curation, P.R.; writing—original draft preparation, P.R. and K.T.; writing—review and editing, P.R. and C.C.; visualization, P.R.; supervision, C.C.; project administration, P.R.; funding acquisition, C.C. All authors have read and agreed to the published version of the manuscript.

Funding: The project was funded in part by National Research Council of Thailand (NCRT, Project No. NRCT5-RSA63002-14).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Public datasets have been used; references are available in the manuscript. The implementation and additional resources are available at https://doi.org/10.6084/m9.figshare.28690706.v1, accesed on 20 March 2025.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Prentzas, J.; Sidiropoulou, M. Assessing the Use of Open AI Chat-GPT in a University Department of Education. In Proceedings of the International Conference on Information, Intelligence, Systems & Applications, Volos, Greece, 10–12 July 2023; pp. 1–4. [CrossRef]
2. Islam, R.; Ahmed, I. Gemini-the most powerful LLM: Myth or Truth. In Proceedings of the 2024 5th Information Communication Technologies Conference (ICTC), Nanjing, China, 10–12 May 2024; pp. 303–308. [CrossRef]
3. Huang, D.; Hu, Z.; Wang, Z. Performance Analysis of Llama 2 Among Other LLMs. In Proceedings of the 2024 IEEE Conference on Artificial Intelligence (CAI), Singapore, 25–27 June 2024; pp. 1081–1085. [CrossRef]
4. Kok, C.L.; Ho, C.K.; Aung, T.H.; Koh, Y.Y.; Teo, T.H. Transfer Learning and Deep Neural Networks for Robust Intersubject Hand Movement Detection from EEG Signals. *Appl. Sci.* **2024**, *14*, 8091. [CrossRef]
5. Reddy, G.P.; Pavan Kumar, Y.V.; Prakash, K.P. Hallucinations in Large Language Models (LLMs). In Proceedings of the 2024 IEEE Open Conference of Electrical, Electronic and Information Sciences, Vilnius, Lithuania, 25 April 2024; pp. 1–6. [CrossRef]

6. Raiaan, M.A.K.; Mukta, M.S.H.; Fatema, K.; Fahad, N.M.; Sakib, S.; Mim, M.M.J.; Ahmad, J.; Ali, M.E.; Azam, S. A Review on Large Language Models: Architectures, Applications, Taxonomies, Open Issues and Challenges. *IEEE Access* **2024**, *12*, 26839–26874. [CrossRef]
7. Vakayil, S.; Juliet, D.S.; Anitha, J.; Vakayil, S. RAG-Based LLM Chatbot Using Llama-2. In Proceedings of the 2024 7th International Conference on Devices, Circuits and Systems (ICDCS), Coimbatore, India, 19–20 April 2024; pp. 1–5. [CrossRef]
8. Epoch AI. Data on Notable AI Models. 2024. Available online: https://epoch.ai/data/notable-ai-models (accessed on 29 March 2025).
9. Thonglek, K.; Takahashi, K.; Ichikawa, K.; Nakasan, C.; Nakada, H.; Takano, R.; Iida, H. Retraining Quantized Neural Network Models with Unlabeled Data. In Proceedings of the 2020 International Joint Conference on Neural Networks (IJCNN), Glasgow, UK, 19–24 July 2020; pp. 1–8. [CrossRef]
10. Stirapongsasuti, S.; Thonglek, K.; Misaki, S.; Usawalertkamol, B.; Nakamura, Y.; Yasumoto, K. A nudge-based smart system for hand hygiene promotion in private organizations. In Proceedings of the Conference on Embedded Networked Sensor Systems. Association for Computing Machinery, Virtual Event, 16–19 November 2020; pp. 743–744. [CrossRef]
11. Achiam, J.; Adler, S.; Agarwal, S.; Ahmad, L.; Akkaya, I.; Aleman, F.L.; Almeida, D.; Altenschmidt, J.; Altman, S.; Gu, S.; et al. GPT-4 Technical Report. *arXiv* **2024**, arXiv:2303.08774.
12. Team, G.; Georgiev, P.; Lei, V.I.; Burnell, R.; Bai, L.; Gulati, A.; Tanzer, G.; Vincent, D.; Pan, Z.; Teplyashin, D.; et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv* **2024**, arXiv:2403.05530.
13. Anthropic. Claude: A Large Language Model by Anthropic. 2024. Available online: https://www.anthropic.com (accessed on 23 January 2025).
14. Mistral AI. Models Overview—Mistral Documentation. 2024. Available online: https://docs.mistral.ai/getting-started/models/models_overview/ (accessed on 21 February 2025).
15. Grattafiori, A.; Dubey, A.; Jauhri, A.; Pandey, A.; Kadian, A.; Al-Dahle, A.; Letman, A.; Mathur, A.; Schelten, A.; Vaughan, A.; et al. The Llama 3 Herd of Models. *arXiv* **2024**, arXiv:2407.21783.
16. Kok, C.L.; Heng, J.B.; Koh, Y.Y.; Teo, T.H. Energy-, Cost-, and Resource-Efficient IoT Hazard Detection System with Adaptive Monitoring. *Sensors* **2025**, *25*, 1761. [CrossRef] [PubMed]
17. Shui, H.; Zhu, Y.; Zhuo, F.; Sun, Y.; Li, D. An Emotion Text Classification Model Based on Llama3-8b Using Lora Technique. In Proceedings of the 2024 7th International Conference on Computer Information Science and Application Technology (CISAT), Hangzhou, China, 12–14 July 2024; pp. 380–383.
18. Hossain, M.B.; Gong, N.; Shaban, M. Computational Complexity Reduction Techniques for Deep Neural Networks: A Survey. In Proceedings of the 2023 IEEE International Conference on Artificial Intelligence, Blockchain, and Internet of Things (AIBThings), Mount Pleasant, MI, USA, 16–17 September 2023; pp. 1–6. [CrossRef]
19. Gao, S.; Huang, F.; Cai, W.; Huang, H. Network Pruning via Performance Maximization. In Proceedings of the 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 20–21 June 2021; pp. 9266–9276. [CrossRef]
20. Ben Letaifa, L.; Rouas, J.L. Fine-grained analysis of the transformer model for efficient pruning. In Proceedings of the 2022 21st IEEE International Conference on Machine Learning and Applications (ICMLA), Nassau, Bahamas, 12–14 December 2022; pp. 897–902. [CrossRef]
21. Chen, P.Y.; Lin, H.C.; Guo, J.I. Multi-Scale Dynamic Fixed-Point Quantization and Training for Deep Neural Networks. In Proceedings of the 2023 IEEE International Symposium on Circuits and Systems, Monterey, CA, USA, 29–31 May 2023; pp. 1–5. [CrossRef]
22. Bao, Z.; Tong, T.; Du, D.; Wang, S. Image Classification Network Compression Technique Based on Learning Temperature-Knowledge Distillation. In Proceedings of the 2024 20th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD), Harbin, China, 29–31 July 2024; pp. 1–7. [CrossRef]
23. Liao, S.; Xie, Y.; Lin, X.; Wang, Y.; Zhang, M.; Yuan, B. Reduced-Complexity Deep Neural Networks Design Using Multi-Level Compression. *Trans. Sustain. Comput.* **2019**, *4*, 245–251. [CrossRef]
24. Abdolali, M.; Rahmati, M. Multiscale Decomposition in Low-Rank Approximation. *IEEE Signal Process. Lett.* **2017**, *24*, 1015–1019. [CrossRef]
25. Yu, Z.; Bouganis, C.S. SVD-NAS: Coupling Low-Rank Approximation and Neural Architecture Search. In Proceedings of the 2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), Waikoloa, HI, USA, 2–7 January 2023; pp. 1503–1512. [CrossRef]
26. Zhang, K.; Liu, G. Layer Pruning for Obtaining Shallower ResNets. *IEEE Signal Process. Lett.* **2022**, *29*, 1172–1176. [CrossRef]
27. Elkerdawy, S.; Elhoushi, M.; Singh, A.; Zhang, H.; Ray, N. One-Shot Layer-Wise Accuracy Approximation For Layer Pruning. In Proceedings of the 2020 IEEE International Conference on Image Processing (ICIP), Virtual, 25–28 October 2020; pp. 2940–2944. [CrossRef]

28. Tang, H.; Lu, Y.; Xuan, Q. SR-init: An Interpretable Layer Pruning Method. In Proceedings of the ICASSP 2023—2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Rhodes Island, Greece, 4–10 June 2023; pp. 1–5. [CrossRef]
29. Ma, X.; Fang, G.; Wang, X. LLM-Pruner: On the Structural Pruning of Large Language Models. In Proceedings of the Advances in Neural Information Processing Systems, New Orleans, LA, USA, 10–16 December 2023; Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., Levine, S., Eds.; Curran Associates, Inc.: Nice, France, 2023; Volume 36, pp. 21702–21720.
30. Liang, J.J.; Qu, B.Y. Large-scale portfolio optimization using multiobjective dynamic mutli-swarm particle swarm optimizer. In Proceedings of the 2013 IEEE Symposium on Swarm Intelligence (SIS), Singapore, 16–19 April 2013; pp. 1–6. [CrossRef]
31. Chu, Z.; Guo, Q.; Wang, C. The PID Control Algorithm based on Whale Optimization Algorithm Optimized BP Neural Network. In Proceedings of the 2023 IEEE 7th Information Technology and Mechatronics Engineering Conference (ITOEC), Chongqing, China, 15–17 September 2023; Volume 7, pp. 2450–2453. [CrossRef]
32. Wong, L.P.; Choong, S.S. A Bee Colony Optimization algorithm with Frequent-closed-pattern-based Pruning Strategy for Traveling Salesman Problem. In Proceedings of the 2015 Conference on Technologies and Applications of Artificial Intelligence (TAAI), Tainan, China, 20–22 November 2015; pp. 308–314. [CrossRef]
33. Sukma Putra, A.; Firman, S.; Srigutomo, W.; Hidayat, Y.; Lesmana, E. A Comparative Study of Simulated Annealing and Genetic Algorithm Method in Bayesian Framework to the 2D-Gravity Data Inversion. *J. Phys. Conf. Ser.* **2019**, *1204*, 012079. [CrossRef]
34. Sun, G.; Shang, Y.; Yuan, K.; Gao, H. An Improved Whale Optimization Algorithm Based on Nonlinear Parameters and Feedback Mechanism. *Int. J. Comput. Intell. Syst.* **2022**, *15*, 38. [CrossRef]
35. Kennedy, J.; Eberhart, R. Particle swarm optimization. In Proceedings of the ICNN'95—International Conference on Neural Networks, Perth, WA, Australia, 27 November 27–1 December 1995; Volume 4, pp. 1942–1948. [CrossRef]
36. Mirjalili, S.; Lewis, A. The Whale Optimization Algorithm. *Adv. Eng. Softw.* **2016**, *95*, 51–67. [CrossRef]
37. Brodzicki, A.; Piekarski, M.; Jaworek-Korjakowska, J. The Whale Optimization Algorithm Approach for Deep Neural Networks. *Sensors* **2021**, *21*, 8003. [CrossRef] [PubMed]
38. Wu, T.; Shi, J.; Zhou, D.; Lei, Y.; Gong, M. A Multi-objective Particle Swarm Optimization for Neural Networks Pruning. In Proceedings of the 2019 IEEE Congress on Evolutionary Computation (CEC), Wellington, New Zealand, 10–13 June 2019; pp. 570–577. [CrossRef]
39. Hu, E.J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; Chen, W. LoRA: Low-Rank Adaptation of Large Language Models. In Proceedings of the International Conference on Learning Representations, Virtual, 25–29 April 2022.
40. EleutherAI. lm-Evaluation-Harness: A Framework for Evaluating Large Language Models. 2021. Available online: https://github.com/EleutherAI/lm-evaluation-harness (accessed on 27 April 2025).
41. Zellers, R.; Holtzman, A.; Bisk, Y.; Farhadi, A.; Choi, Y. HellaSwag: Can a Machine Really Finish Your Sentence? In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, Florence, Italy, 28 July–2 August 2019; pp. 4791–4800. [CrossRef]
42. AI, M. Llama 3.1 70B Instruct Model. 2024. Available online: https://huggingface.co/meta-llama/Llama-3.1-70B-Instruct (accessed on 20 February 2025).
43. Le, V. Mealpy: A Python Metaheuristic Optimization Librar. 2024. Available online: https://mealpy.readthedocs.io/en/latest/ (accessed on 3 November 2024).