Optimization and Segmentation of Large Language Models on Edge Devices

Dr. Rashmi Adyapady R.
Dept. of AIML
NMAM Institute of Technology
Nitte, India
rashmi.adyapady@nitte.edu.in

Chethana R Kini
Dept. of AIML
NMAM Institute of Technology
Nitte, India
chethanarkini@gmail.com

Manya Hegde
Dept. of AIML
NMAM Institute of Technology
Nitte, India
hegdemanya2003@gmail.com

Reyona Elza Sabu
Dept. of AIML
NMAM Institute of Technology
Nitte, India
reo.elza@gmail.com

***Abstract*—Large language models (LLMs) have transformed natural language tasks like text generation, translation, and summarization. However, their large size and high processing needs make them difficult to deploy on devices with limited memory and power, such as mobile phones and IoT devices. This study tackles these challenges by applying optimization techniques to make LLMs more efficient for such devices. Methods like model pruning reduce the model size, while evolutionary algorithms such as Particle Swarm Optimization (PSO) and Genetic Algorithms (GA) enhance performance. Segmentation through pipeline parallelism further allows different parts of the model to run across multiple devices, saving time and energy. These results show that LLMs can be optimized for resource-limited settings, enabling new AI applications on edge devices.**

***Keywords*—Large Language Models (LLMs), Natural Language Processing (NLP), Edge Devices, Model Optimization, Model Pruning, Evolutionary Algorithms, Particle Swarm Optimization (PSO), Genetic Algorithms (GA), Pipeline Parallelism, Model Segmentation, Distributed Computing**

I. Introduction

Large Language Models (LLMs) are advanced artificial intelligence systems designed to understand and generate human language, supporting tasks such as text generation, translation, and summarization. These models rely on vast datasets and deep neural architectures to produce coherent, contextually relevant language output, revolutionizing the way machines process human language. Despite their potential, deploying LLMs on smaller devices, such as smartphones, Internet of Things (IoT) devices, and embedded systems, remains a challenge due to the substantial computational power and memory they require. Edge devices, with their limited processing capabilities and energy constraints, often depend on cloud-based solutions to handle these intensive AI tasks, which can introduce delays, increase costs, and raise privacy concerns.

This study focuses on developing methods to optimize and segment LLMs for efficient deployment on edge devices, enabling localized, real-time language processing without relying on cloud resources. To achieve this, we employ evolutionary algorithms, including Particle Swarm Optimization (PSO) and Genetic Algorithms (GA), to fine-tune the model parameters for better processing efficiency while preserving performance. Additionally, model pruning techniques, such as L1 pruning, help reduce the model's size by removing less significant elements without compromising accuracy. Segmentation via pipeline parallelism divides the model into smaller segments that can be distributed across multiple devices, balancing the computational load and improving response times.

Our approach demonstrates the feasibility of adapting LLMs for edge devices, allowing applications in areas like healthcare, smart home technology, and autonomous systems to operate effectively with local AI processing. By eliminating the need for continuous cloud connectivity, this research enhances the speed, privacy, and accessibility of LLM-powered applications across diverse environments, paving the way for broader and more secure usage.

II. Motivation

The increasing demand for advanced AI capabilities in real-time applications, especially on devices with limited resources, motivated this project. Large language models (LLMs) offer impressive performance in tasks like text generation and translation but are too large and computationally expensive to run on small devices such as smartphones, IoT devices, and embedded systems. These devices have limited memory, processing power, and energy, making it difficult to deploy resource-heavy models efficiently.

Optimizing LLMs for edge devices will not only improve their accessibility in areas with limited infrastructure but also

reduce reliance on cloud servers, lowering costs, improving response times, and enhancing privacy. This project aims to bridge the gap between powerful AI models and resource-constrained environments, enabling AI-driven solutions in fields such as healthcare, smart cities, and mobile applications.

III. Literature Survey

Large Language Models (LLMs) present transformative capabilities in Natural Language Processing (NLP) tasks, yet deploying these models on edge devices introduces significant challenges due to constraints on memory, processing power, and energy. This survey focuses on the latest advancements in optimizing LLMs for edge applications, employing techniques like Evolutionary Algorithms (EAs), pipeline parallelism, and model segmentation.

Integrating large language models (LLMs) with evolutionary algorithms (EAs) has been shown to enhance optimization tasks, where EAs efficiently explore complex search spaces and LLMs contribute to improving solution quality through informed guidance. This approach has demonstrated effectiveness in areas such as code generation and other AI applications, but requires further refinement to address the challenges of high-dimensional optimization problems effectively [1].

For neural network optimization, a hybrid Particle Swarm Optimization (PSO) and Cuckoo Search (CS) algorithm has been proposed, which combines PSO with CS to achieve a balance between fast convergence and robust global optimization. The PSO-CS method demonstrates superior accuracy in training Feedforward Neural Networks (FNNs), achieving 94.8% accuracy on the Iris dataset, outperforming traditional PSO and CS methods [2].

Dynamic Weight Particle Swarm Optimization (DWPSO) has been introduced for optimizing Support Vector Machines (SVMs). By adapting the inertia weight dynamically, DWPSO mitigates the risk of premature convergence and local minima, improving classification accuracy on diverse datasets such as Diabetes and Heart Disease. The effectiveness of DWPSO highlights the potential of adaptive optimization in enhancing the performance of large language models (LLMs) on constrained devices [3].

PipeDream [4] leverages pipeline parallelism for Distributed Neural Network (DNN) training, significantly reducing communication overhead and improving throughput. Harlap et al. demonstrate that by partitioning DNNs across multiple GPUs and enabling asynchronous communication, PipeDream achieves up to fivefold performance improvements over data-parallel methods, making it a viable solution for deploying large models in resource-constrained environments.

Pipeline parallelism is explored in the GPipe framework, which scales large models by distributing model layers across multiple devices. It uses gradient accumulation and micro-batching to handle memory constraints and facilitate gradient updates. This approach reduces communication bottlenecks during training. GPipe supports models with billions of parameters, enabling efficient model training at scale [5].

IV. Proposed Model

The proposed model consists of four primary stages designed to optimize and deploy Large Language Models (LLMs) on edge devices, addressing their inherent constraints in computational power, memory, and energy resources.

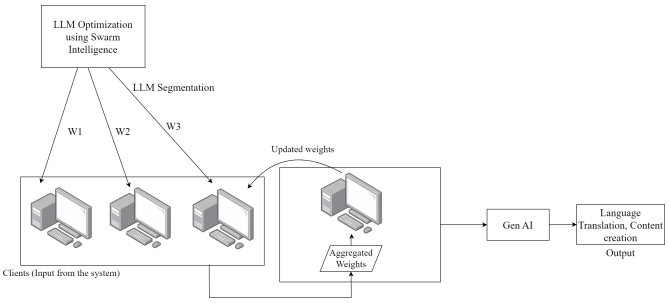

Fig. 1. Proposed Model

The first step focuses on optimizing LLMs through evolutionary algorithms, including Particle Swarm Optimization (PSO) and Genetic Algorithms (GA). These algorithms are utilized to fine-tune the model parameters, which reduces the overall computational requirements, memory usage, and energy consumption. This optimization step is critical for making the models suitable for environments with limited resources, such as edge devices.

Following optimization, model pruning techniques are applied to further reduce the size of the model by removing parameters deemed less significant. Techniques like L1 pruning streamline the model without compromising accuracy, resulting in a more lightweight and efficient model that is better suited for real-time inference on edge devices.

Once the model has been pruned, it is segmented into smaller, manageable components that can be distributed across multiple edge devices. Each device processes a specific segment of the model independently, enabling parallel execution. This segmentation strategy ensures effective utilization of distributed computational resources, helping to minimize latency and improve processing throughput.

The final step involves aggregating the outputs from each segment to generate a unified result. This aggregation ensures that the segmented model functions as a cohesive unit, providing accurate and efficient results for various tasks, such as language translation, content generation, or summarization.

V. System Design

This project enables Large Language Models (LLMs) to operate efficiently on resource-limited edge devices through a streamlined process of fine-tuning, optimization, pruning, and segmentation.

(i) *Fine-Tuning*: The LLM is initially fine-tuned with a specific dataset to enhance its performance for targeted tasks.

(ii) *Optimization*: After fine-tuning, evolutionary algorithms such as Particle Swarm Optimization (PSO) and Genetic

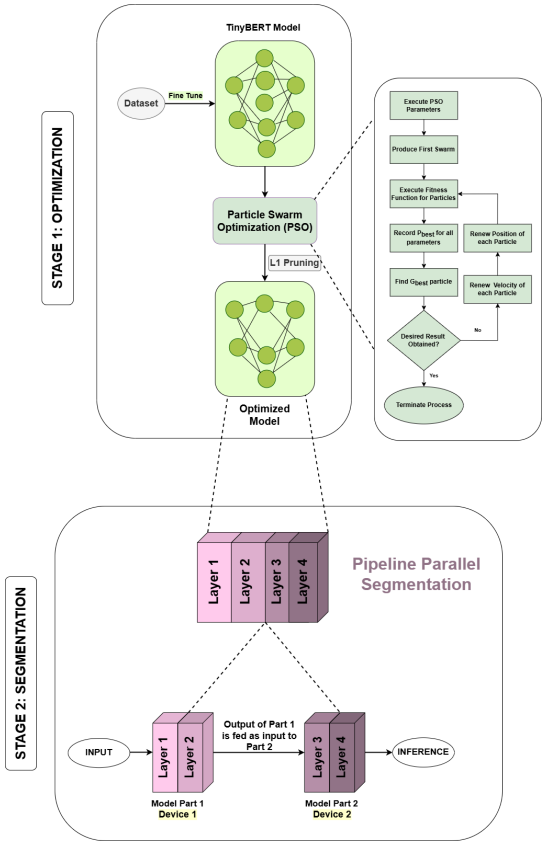

Fig. 2. System Design

Algorithm (GA) are applied to adjust model hyperparameters, reducing computational and memory requirements.

(iii) *Pruning*: Following optimization, L1 pruning techniques are used to remove less significant parameters, minimizing the model size and making it more efficient for inference.

(iv) *Segmentation*: Finally, the optimized and pruned model is segmented and distributed across multiple edge devices. Each device processes a portion of the model, enabling efficient parallel execution within resource constraints.

This systematic approach ensures that LLMs are both lightweight and capable of real-time processing on edge devices.

VI. System Implementation

The implementation of this methodology enables the deployment of Large Language Models (LLMs) on edge devices by utilizing optimization, pruning, and segmentation techniques. The following sections provide details on dataset preparation, model description, optimization methods, and the two approaches used in pipeline parallelism.

A. Dataset Description

The IMDB dataset is used for fine-tuning and evaluating the model. This dataset contains 50,000 movie reviews, evenly split between positive and negative sentiments. It is commonly used for binary sentiment classification tasks. The dataset provides a balanced distribution of positive and negative samples, making it ideal for training models to recognize sentiment patterns in text.

B. Model Description

The implementation uses a pre-trained TinyBERT model from the Hugging Face Transformers library. TinyBERT is a distilled, lightweight variant of the BERT model, specifically designed to retain high accuracy while minimizing memory and computational requirements. Fine-tuning is performed on the IMDB dataset to adapt TinyBERT to the sentiment classification task, enhancing its performance on the specific domain of movie reviews.

C. Optimization Using Evolutionary Algorithms

Evolutionary algorithms, namely Particle Swarm Optimization (PSO) and Genetic Algorithm (GA), are applied to optimize TinyBERT's hyperparameters. These techniques iteratively adjust the model to reduce computational demands, enhancing its suitability for edge deployment. PSO leverages a swarm of particles (candidate solutions) to explore the hyperparameter space, while GA applies selection, crossover, and mutation to evolve configurations that yield improved performance and resource efficiency.

D. Pruning

After optimization, the model is pruned using L1 pruning to remove low-weight parameters, reducing the model's size and computational complexity. This step ensures that the model can perform efficiently on memory-constrained edge devices while retaining essential features and maintaining high accuracy.

E. Pipeline Parallel Segmentation

The pruned model is segmented to enable pipeline parallelism, reducing latency and optimizing the use of multiple devices. Two approaches are explored for this segmentation:

1) Approach 1 (Multi-Device Pipeline Parallelism): The optimized TinyBERT model is split into two parts, each deployed on a separate device:

- Device 1 processes the initial layers of the model, handling data input and performing preliminary computations. The intermediate output is then sent to the next device.
- Device 2 completes the remaining layers, producing the final output directly from its computation. This configuration leverages pipeline parallelism across two devices, enhancing throughput and allowing for efficient distributed processing on resource-constrained edge devices.

2) Approach 2 (Single-Device Pipeline Parallelism with Microbatches): In this approach, the optimized TinyBERT model operates on a single device, where pipeline parallelism is implemented with asynchronous mini-batches, allowing for overlapping computations:

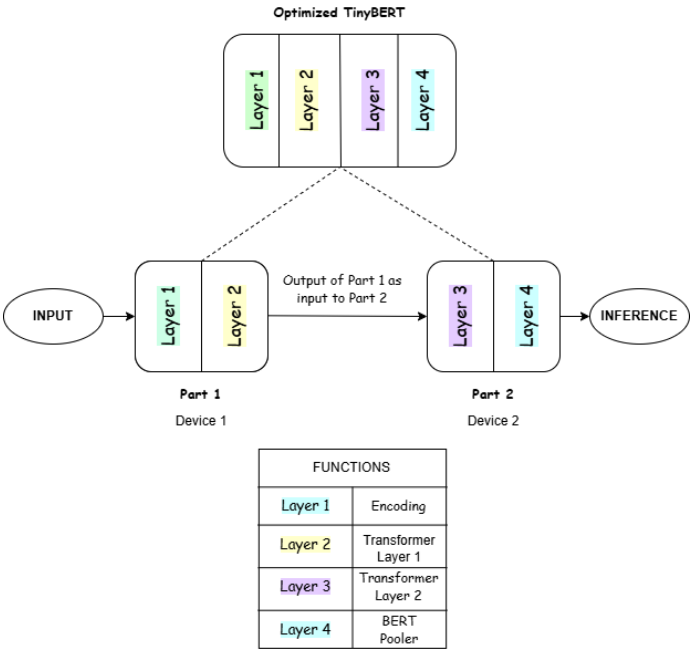

Fig. 3. Approach 1

- The model is split into two segments, with each segment processing a microbatch in sequence. As the first segment finishes processing one microbatch, it sends the output to the next segment and starts on the subsequent microbatch. This overlap maximizes processing speed by ensuring that the device is continuously utilized without idle time.
- This approach is suited for scenarios where only a single device is available, but there is a need to maximize computational efficiency. By reducing idle time, this configuration maintains high processing speeds, even on limited hardware resources.

F. Inter-Device Communication Using Socket Programming

For multi-device configurations (Approach 1), socket programming is employed to facilitate data transfer between devices. This communication protocol allows intermediate outputs from one device to be passed seamlessly to the next, ensuring efficient data flow throughout the pipeline. Python's socket library is used to implement the client-server model, where each device either receives input from or sends output to its neighboring device.

VII. Results and Analysis

In this study, several optimization techniques and segmentation approaches were applied to different model architectures to enhance their performance on edge devices. This section presents the results achieved in terms of model accuracy, processing time, and efficiency improvements after applying various optimization methods.

A. Optimization Results

1) **Genetic Algorithm (GA):** GA was employed to optimize the model weights of Artificial Neural Networks (ANNs) and Convolutional Neural Networks (CNNs) using the IMDB dataset. The results showed that ANN achieved an accuracy of 56% while CNN reached 51%, demonstrating the Genetic Algorithm's limited success on these architectures (Table I).

TABLE I
Performance of Genetic Algorithm

Model	Accuracy
ANN	56%
CNN	51%

2) **Bat Algorithm (BA):** The Bat Algorithm, based on simulating echolocation, provided competitive accuracy with the ANN model reaching 85% (Table II). This suggests the BA was effective for optimizing ANN on the IMDB dataset.

TABLE II
Performance of Bat Algorithm

Model	Accuracy
ANN	85%

3) **Particle Swarm Optimization (PSO):** PSO, a social behavior-based optimization technique, was effective in improving model performance. ANN achieved an accuracy of 87.49%, CNN reached 74%, SVM recorded 88.7%, and TinyBERT attained 85% accuracy, indicating PSO's robustness across model types (Table III).

TABLE III
Performance of Particle Swarm Optimization

Model	Accuracy
ANN	87.49%
CNN	74%
SVM	88.7%
TinyBERT	85%

4) **Hybrid Approach:** Hybrid optimization combining PSO with other algorithms showed notable results. PSO combined with Adam optimization yielded 85.34% accuracy on ANN, while PSO combined with Genetic Algorithm reached 85.42% accuracy (Table IV).

TABLE IV
Performance of Hybrid Approach

Model	Technique	Accuracy
ANN	PSO + Adam	85.34%
ANN	PSO + GA	85.42%

B. Segmentation Results

1) **Approach 1:** In the first segmentation approach, the optimized TinyBERT model was divided into two parts for client-server deployment. The client processed text reviews with the initial model layers and passed results to the server, which completed the final classification. This setup demonstrated improved efficiency, handling

memory constraints effectively and achieving the target of classifying sentiment in text reviews.

2) **Approach 2:** The second segmentation approach used pipeline parallelism with micro-batching, enabling overlapping computations between devices for improved resource utilization. The accuracy was consistent with Approach 1 at 78.8%, but this approach increased processing speed and efficiency by maximizing parallelism.

C. Analysis of Results

TABLE V
COMPARISION OF ALGORITHMS

Algorithm	Model	Accuracy
GA	ANN	56%
PSO	ANN	87.49%
BAT	ANN	85%
PSO + GA	ANN	85.42%

The results, as indicated in Table V, show that Particle Swarm Optimization (PSO) was the most effective method for optimizing model performance, particularly in achieving the highest accuracy with the ANN model at 87.49%. Hybrid approaches also demonstrated effectiveness, with the combination of PSO and Genetic Algorithm (PSO + GA) achieving the highest ANN accuracy at 85.42%.

In terms of segmentation, both pipeline approaches proved effective in distributing model processing across devices, enhancing real-time processing capabilities while reducing latency. The pipeline parallelism with micro-batching showed further improvements in computation-communication overlap, making it suitable for scalable deployment on edge devices with constrained resources.

VIII. CONCLUSION AND FUTURE WORK

A. Conclusion

This research explores the optimization of Large Language Models (LLMs) for deployment on edge devices by employing various evolutionary and segmentation techniques. Through methods like Particle Swarm Optimization (PSO), Genetic Algorithms (GA), Bat Algorithm, and hybrid approaches, significant improvements were observed in model accuracy and resource efficiency. Among these, PSO proved to be the most effective, achieving high accuracy rates across models such as ANN, SVM, and TinyBERT. The hybrid approaches further enhanced optimization results, with PSO combined with GA achieving the highest accuracy of 85.42% for ANN models.

To address resource constraints on edge devices, segmentation methods were applied to distribute model processing across multiple devices. Pipeline parallelism, both with and without micro-batching, successfully enabled efficient model deployment by reducing memory usage and latency while maintaining accuracy. The micro-batching approach in pipeline parallelism showed additional improvements in processing speed by maximizing resource utilization and overlap between computation and communication tasks.

This study demonstrates the potential of these optimization and segmentation techniques in making advanced LLMs feasible on edge devices, such as smartphones, IoT devices, and embedded systems, where memory and processing power are limited. The successful deployment of optimized models opens new possibilities for real-time AI applications in mobile environments, healthcare, autonomous systems, and other fields where edge computing is essential.

B. Future Work

Future research aims to explore advanced model partitioning strategies for LLMs on edge devices. This could involve task-specific segmentation based on computational power and resource constraints of each device, enabling even more efficient scaling. Additionally, applying mathematical optimization techniques to further balance load distribution across devices could improve performance. Expanding the current segmentation techniques to support more than two devices and incorporating real-time monitoring to dynamically adjust segmentation could further enhance scalability and adaptability of LLMs in diverse computing environments.

REFERENCES

[1] X. Wu, S. H. Wu, J. Wu, L. Feng, and K. C. Tan, "Evolutionary computation in the era of large language model: Survey and roadmap," *arXiv preprint arXiv:2401.10034*, 2024.

[2] J. F. Chen, Q. H. Do, and H. N. Hsieh, "Training artificial neural networks by a hybrid PSO-CS algorithm," *Algorithms*, vol. 8, no. 2, pp. 292-308, 2015.

[3] J. Wang, X. Wang, X. Li, and J. Yi, "A hybrid particle swarm optimization algorithm with dynamic adjustment of inertia weight based on a new feature selection method to optimize SVM parameters," *Entropy*, vol. 25, no. 3, p. 531, 2023.

[4] A. Harlap, D. Narayanan, A. Phanishayee, V. Seshadri, N. Devanur, G. Ganger, and P. Gibbons, "Pipedream: Fast and efficient pipeline parallel DNN training," *arXiv preprint arXiv:1806.03377*, 2018.

[5] Y. Huang, Y. Cheng, A. Bapna, O. Firat, M. X. Chen, D. Chen, ... and Z. Chen, "GPipe: Easy scaling with micro-batch pipeline parallelism," in *Proceedings of Computer Science: Computer Vision and Pattern Recognition*, 2019.

[6] A. Imteaj, U. Thakker, S. Wang, J. Li, and M. H. Amini, "A survey on federated learning for resource-constrained IoT devices," *IEEE Internet of Things Journal*, vol. 9, no. 1, pp. 1-24, 2021.

[7] A. O. Fajemisin, D. Maragno, and D. den Hertog, "Optimization with constraint learning: A framework and survey," *European Journal of Operational Research*, vol. 314, no. 1, pp. 1-14, 2024.

[8] S. Feng, Y. Chen, Q. Zhai, M. Huang, and F. Shu, "Optimizing computation offloading strategy in mobile edge computing based on swarm intelligence algorithms," *EURASIP Journal on Advances in Signal Processing*, vol. 2021, pp. 1-15, 2021.