NITTE EDUCATION TRUST

N.M.A.M. INSTITUTE OF TECHNOLOGY
(An Autonomous Institution affiliated to Visvesvaraya Technological University, Belagavi)
Nitte – 574 110, Karnataka, India

**Department of Artificial Intelligence and Machine Learning**

**Project Phase I-Title Acceptance and Literature Review**

**Subject: Project Lab** **Subject Code: 21AM704**

# SwarmBee: Swarm Intelligence meets GenAI on Edge

**Team Members**
1. Deepshika Poojary [4NM21AI022]
2. Saachi Suvarna [4NM21AI061]

**Dr. Sharada U Shenoy**
Professor & Head
Department of AIML
NMAMIT-Nitte

## Contents

## Introduction

The rising popularity of LLMs attracted widespread attention from the public, industry, and academia, leading to their application to various domains. Compared to previous language models, LLMs (e.g., GPT-4, LLaMA, Claude, Gemini) show much greater context and generalization and memory capabilities. Hence, they are used in tasks like text generation, translation and question-answering.

Many enterprises deploy these models on cloud as they offer benefits such as scalability, efficiency, and advanced computational capabilities.

However, the heavy reliance of these LLMs on cloud computing, leads to latency, high bandwidth cost, and privacy issues. A way to combat these challenges is to deploy these models on the devices at the network edge, closer to the data sources.

But, deploying these models on the edge devices has its own concerns. LLMs are resource-intensive, i.e., they require significant computational power and memory, which are major limiting factors in edge devices.

## Motivation and Scope

Deploying LLMs on edge devices answers latency, connectivity and data privacy issues. But, the significant memory and computational costs incurred during the inference process make it challenging to deploy large models on these resource-constrained devices.

Some prominent solutions are Model Quantization, Model Pruning, Knowledge Distillation, Federated Learning, Swarm Optimization and Cloud-Edge Collaboration.

A solution that stands out is Federated learning. It allows for the collaborative training of LLMs across multiple decentralized nodes without the need to share their actual data. This approach preserves the privacy and security of the data and harnesses the collective intelligence of the swarm. It also enhances the robustness and adaptability of the models, making them capable of generalizing across diverse datasets.

The aim of the research is to review the existing literature and address the challenges faced in integrating federated LLMs with swarm principles.

# Compressing Large Language Models by Streamlining the Unimportant Layers

As the number of parameters of the models increases it leads to raising hardware requirements, which significantly constrains their applicability and hinders their deployment in real-world scenarios.

1. Model quantization compresses the model by modifying the parameter storage format, however, it does not reduce the number of parameters.
2. Model distillation, as opposed to quantization, compresses model parameters but comes with a substantial computational and storage cost.
3. Model pruning can more effectively compress model parameters, but existing pruning techniques usually result in irregular model structures, which makes them inconvenient for practical use.

## Proposed Method:

1. **Layer Pruning:** Here a set of consecutive layers with the lowest importance in the model according to the target sparsity. The importance of layers based on the cosine similarity between the hidden states of their inputs and outputs and prunes those layers that are deemed less important.
2. **Layer Replacement:** Here a lightweight model is trained to substitute the pruned layers, thereby mitigating the performance degradation caused by pruning.

The lightweight models used to replace the pruned layers in the study include multi-layer perceptrons (MLPs), transformer layers, sharing-parameter MLPs, and sharing-parameter transformer layers. The experiments demonstrated that **a single MLP can effectively fit the pruned layers**, outperforming the other lightweight model configurations in terms of performance while being more parameter-efficient.

## Results:

The experiments conducted in the study were performed on the following large language models (LLMs):

1. OPT-1.3B
2. OPT-2.7B
3. OPT-6.7B
4. Llama2-7B

Experiments have shown that using only an MLP to replace multiple consecutive transformer layers can maintain up to 92% of the LLM’s classification performance and 68% of the LLM’s generation performance while reducing the model’s parameter count to about 25%, outperforming previous state-of-the-art (SOTA) pruning methods.

## Limitations:

1. The model obtained from pruning still has a large performance gap with the model that satisfies the scaling law.
2. The method has a large performance improvement compared to the baseline method, it still has a large performance loss compared to the original model.

# EdgeShard: Efficient LLM Inference via Collaborative Edge Computing

The paper proposes a general framework to partition the LLM model into shards and deploy on distributed devices.

To achieve efficient LLM inference, M. Zhang et al. have designed a dynamic programming algorithm to optimize the inference latency and throughput.

The framework has three stages:

1. Profiling: The following traces are profiled:
   i) execution time of each player on different devices
   ii) size of activations and memory consumption for each layer of the LLM model
   iii) available memory of each device and the bandwidth among devices

# EdgeShard: Efficient LLM Inference via Collaborative Edge Computing

2. Scheduling Optimization: The scheduler generates a deployment strategy by determining which device to participate in, how to partition the LLM model in a layer wise, and which device should the model shard be allocated to.

3. Collaborative Inference: Two cases are considered for the collaborative inference, i.e., sequential inference and pipeline parallel inference.

   i) In sequential inference, devices take turns to perform the computation with the allocated model shards.

   ii) For the pipeline parallel inference, the input data will first be split into micro-batch and subsequently feed into the system.

## EdgeShard: Efficient LLM Inference via Collaborative Edge Computing

The proposed method is compared with the following baseline models:

1. Edge-Solo: LLMs are deployed locally on an edge device without model partition.
2. Cloud-Edge-Even: LLMs are evenly partitioned into two parts. One is allocated to the edge device, and another is allocated to the cloud server.

**Result**

EdgeShard achieves lower inference latency and higher inference throughput than baseline methods. For Llama2-7B model, EdgeShard achieves 75.88ms latency, which is about 1.85x faster than Edge-Solo, and about 3x faster than Cloud-Edge-Even.

For the inference throughput, EdgeShard achieves 52.45 tokens per second with a maximum batch size of 8, which is around 2.2 times larger than Edge-Solo, and about 7 times larger than Cloud-Edge-Even.

## Limitations

1. If the devices (to which the multiple shards are to be allocated) belong to different stakeholders, they may not be willing to share devices' computation resources.
2. The latency between edge devices and the central cloud is usually high and unstable, which will affect the inference and fine-tuning performance of LLM.
3. Due to the reduced memory usage of the edge devices, larger batch devices may be allowed, leading to increased throughput. However, the designed dynamic programming algorithm does not consider the influence of batch size.

## Edge Intelligence Optimization for Large Language Model Inference with Batching and Quantization

1. Transformer-based LLMs are significantly larger in size. This increases both processing time and memory consumption.
2. Transformer decoder-based LLMs generate outputs sequentially in an autoregressive manner . Given an input sequence, LLM traverses the entire layer stack to produce only one output token. This procedure is repeated for each output token, intensifying the computational demands.
3. LLMs employ self-attention modules, which means each output token is computed based on all preceding tokens. Consequently, the memory cache for all prior tokens must be maintained until the output sequence concludes, further increasing memory demands.

## Proposed Methods

1. For edge intelligence optimization problem for LLM inference in wireless networks, aiming to maximize inference throughput via batching scheduling.
2. Constraints include communication and memory resources on edge devices and user-specific latency and accuracy demands.The problem is a variant of multidimensional knapsack problem, which is NP-hard.
3. An optimal Depth-First Tree-Searching algorithm with tree-Pruning (DFTSP) to address the problem in a polynomial time. The solution space is constructed as a search tree. To expedite the solution process, searching prioritizes branches formed by requests with higher latency tolerance, shorter output length, and lower communication bandwidth demands.

## Results:

1. The simulation demonstrates the superior throughput performance of DFTSP against other batching schemes across various user settings and quantization methods.

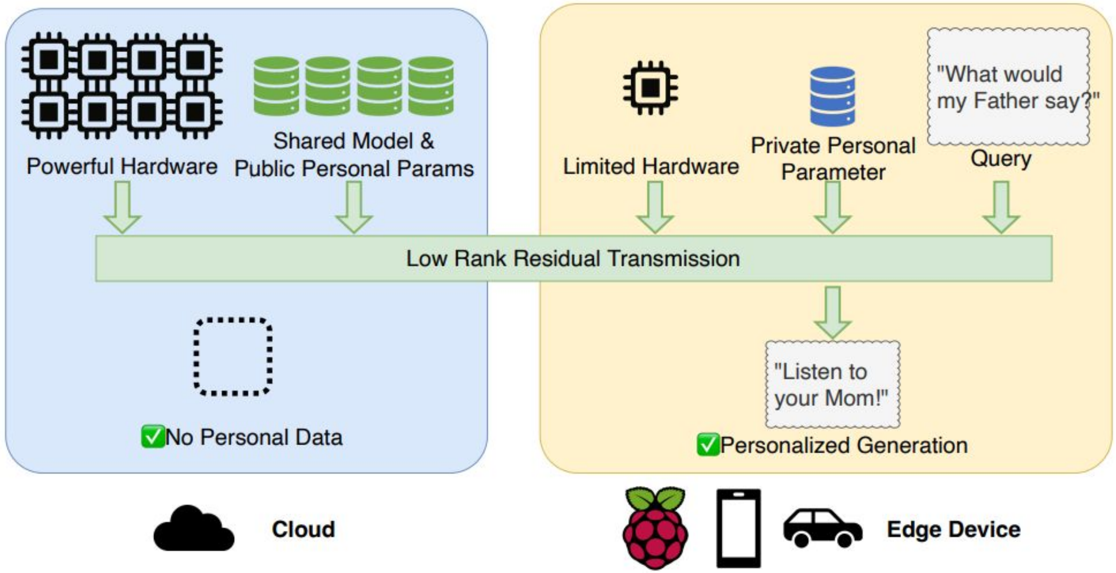
2. Simulation results indicate that DFTSP surpasses other batching benchmarks in throughput across diverse user settings and quantization techniques, and it reduces time complexity by over 45% compared to the brute-force searching method.

## PrivateLoRa for Efficient Privacy Preserving LLM

The paper proposes a LLM service paradigm that distributes privacy-sensitive computation on edge devices and shared computation in the cloud. Only activations are transmitted between the central cloud and edge devices to ensure data locality.

Raw data and personalized parameters M are kept on edge devices throughout training and inference. Only unreadable activations and gradients are transmitted. Thanks to Low Rank Residual Transmission, the communication overhead of activations are reduced by over 95% percent, yielding comparable throughput to cloud-only solutions.

# PrivateLoRa for Efficient Privacy Preserving LLM


Powerful Hardware
Shared Model & Public Personal Params
Limited Hardware
Private Personal Parameter
"What would my Father say?"
Query
Low Rank Residual Transmission
"Listen to your Mom!"
✅No Personal Data
✅Personalized Generation
Cloud
Edge Device

(b) PrivateLoRA-powered paradigm

## PrivateLoRa for Efficient Privacy Preserving LLM

**Result**

PrivateLoRA aims to achieve good performance, parameter-based personalization and data locality. It personalizes the LLMs by tuning knowledge or preferences into the models. It also drastically reduces local compute demands, making it compatible with edge devices.

PrivateLoRA achieved 175.5 and 26.5 tokens per second on 7B model for generation prefill and decoding, respectively, which are both over 300% of device-only solutions.

**Future Work**

1. Redundancy in adaptation of decoder layers can be exploited to further reduce communication overhead.
2. A method should be found to completely poison the activations so that even semantics can not be deduced.

# Observation

| Title | Authors (Year) | Proposed Methodology | Result | Limitations/Future Work |
|---|---|---|---|---|
| Compressing large language models by streamlining the unimportant layer | Chen Xiaodong, Yuxuan Hu, and Jing Zhang (May 2024). | Proposes layer pruning where less important layers are found using cosine similarity and are replaced using a single multilayer perceptron. | 92% of the LLM's classification and 68% of generation performance and 25% reduction in parameters. | Large performance gap as compared to the model. |
| Edge Intelligence Optimization for Large Language Model Inference with Batching and Quantization | Zhang Xinyuan, Jiang Liu, Zehui Xiong, Yudong Huang, Gaochang Xie, and Ran Zhang (May 2024). | Model Quantization is carried to reduce memory footprint. Batch processing is done to process requests from multiple users in parallel | Better throughput across diverse user and quantization, it reduces time complexity by over 45% compared to the brute-force searching . | |
| EdgeShard: Efficient LLM Inference via Collaborative Edge Computing | Zhang Mingjin, Jiannong Cao, Xiaoming Shen, and Zeyang Cui (May 2024). | Proposed a framework that partitions the LLM model into shards, deploys them on distributed devices and enables collaborative inference among heterogeneous edge devices and cloud servers. | Lower inference latency and higher inference throughput than device-only solutions and cloud-edge-even solutions | ● Reduce latency between edge devices and the central cloud. ● Amend the dynamic programming algorithm to consider the influence of batch size. |
| PrivateLoRa for Efficient Privacy Preserving LLM | Wang Yiming, Yu Lin, Xiaodong Zeng, and Guannan Zhang (Nov. 2023). | Distributes privacy-sensitive computation on edge devices and shared computation in the cloud. Only activations are transmitted to ensure data locality. | Reduced local compute demands, Data Locality, thrice the throughput of device-only solutions. | Decrease redundancy in adaptation of decoder layers to further reduce communication overhead. |

## References

[1] Chen, Xiaodong, Yuxuan Hu, and Jing Zhang., "Compressing large language models by streamlining the unimportant layer." arXiv preprint arXiv:2403.19135, May 2024.

[2] Zhang X., Liu, J., Xiong, Z., Huang, Y., Xie, G., and Zhang, R. "Edge Intelligence Optimization for Large Language Model Inference with Batching and Quantization." arXiv preprint arXiv:2405.07140, May 2024.

[3] Zhang M., Cao J., Shen X., Cui Z., "EdgeShard: Efficient LLM Inference via Collaborative Edge Computing." arXiv preprint arXiv:2405.14371, May 2024.

[4] Wang, Y., Lin, Y., Zeng, X., & Zhang, G. “PrivateLoRA For Efficient Privacy Preserving LLM” CoRR, vol. abs/2311.14030, Nov. 2023.