A Comprehensive Analysis on Edge Deployment and Optimization of Llms

Kaibo Liang *

Department of Electrical and Computer Engineering, The University of British Columbia, Vancouver, BC, Canada

* Corresponding Author Email: liang21k@student.ubc.ca

Abstract. After years of development, impressive capabilities are demonstrated by various large language models (LLMs) across many fields. Traditionally, cloud appears to be the preferred deployment option for most applications. The remarkable reasoning performance relies on the support of large and continuous computational power. Increased latency, bandwidth consumption overheads, and lack of privacy protections. All of these are the costs that come with cloud deployment and causing it unsuitable for certain scenarios. In order to address these challenges, increasing research interests toward localized deployment of LLMs have prompted. Such deployment also introduces several issues including limited computational capability, memory capacity, and power consumption of edge devices. This paper first provides an overview of different LLM architectures. Then introduces several proposed approaches aimed at addressing the issues mentioned. Multiple directions are covered, including algorithm-level techniques such as quantization, lightweight or distilled models, and MoE-based architectures; hardware acceleration optimization; algorithm–hardware co-design approaches; and edge–cloud collaboration like split inference strategies. A range of studies are referenced to provide evidence and analysis for the respective approaches. Together it provides insights into current research status, difficulties, and potential opportunities in the related field.

Keywords: LLMs; edge deployment; quantization; light-weight models; edge-cloud collaboration.

1. Introduction

Machine learning (ML), this concept was introduced decades ago but is undergoing continuous advancement in recent years. The Transformer architecture is brought into the spotlight with the introduction of self-attention mechanism, attracting attention from both academia and industry. It has raised a wide range of applications, large language models (LLMs) are the most popular example [1]. As these applications become increasingly embedded into people's daily lives, expectations of their performance, reliability, and accessibility from users are continuously rising.

Majority of current LLM applications rely heavily on cloud-based computation. Strong performance however does come with several challenges [2]. For application scenarios like autonomous driving, robotics, and wearable life-monitoring devices where real-time inference is critical. Frequent communication with the cloud increases the dependency on a stable network connection and inevitably introduces additional latency. Diverse input data types such as text, images, audio, and video and the growing volume of input sequences have placed considerable stress on transmission bandwidth between edge devices and cloud infrastructure. Data privacy has also started to concern individual users as the applications are increasingly integrated into daily life. Beyond individual users, institutions such as banks, financial organizations, and law firms also impose extremely high requirements on their data privacy. However, cloud-based computation does not always provide complete assurance of data confidentiality and privacy preservation. Consequently, these challenges have prompted increasing research interest towards localized deployment of LLMs. The deployment of LLMs on edge devices, commonly referred to as LLM on edge, has emerged as a rapidly growing and widely discussed research topic in recent years.

In response to this issue, extensive efforts from both academia and industry have proposed a wide range of approaches to address this challenge from different perspectives. These efforts include adjustments at model level, as well as developing collaborative edge-cloud frameworks to balance

performance and resource utilization. Furthermore, advancements in hardware design have led to continuously improving the computational capabilities of edge devices, which also contributes to help mitigating these problems to some extent. The remainder of this paper is organized as follows. Section 2 presents LLM fundamentals and review existing approaches. Section 3 analyze these approaches in detail and raise discussions and recommendations. Finally, section 4 concludes the paper with a summary of findings and future directions.

2. Theory Analysis

A LLM is a branch of deep learning system using the transformer architecture, training on massive datasets to predict and generate human-like language. An LLM models the probability distribution of words or tokens in context, allowing it to understand and produce coherent, contextually relevant language across diverse domains. The classical Transformer architecture consists of two main components: the encoder and the decoder. The encoder tokenizing the input sequence, understanding the relationships among all input tokens by using the self-attention mechanism. The input tokens are then used to generate output sequence along with the previously generated tokens by the decoder module. Originating from the basic encoder-decoder architecture, other variants including encoder-only and decoder-only architecture are tailored for distinct use cases.

2.1. Encoder-only Architecture

The name of the encoder-only architecture suggests its architecture, only the encoder module is included in this type of model. A typical encoder model usually consists of two main components: input embedding module and a stack of encoder layers. The embedding module converts the input sequence into embedding vectors. Each encoder layer is identical in structure, each made up of two submodules: a multi-head self-attention module that determines the relationships and dependencies among all tokens in the input sequence, nonlinear transformation is then applied to each token independently by a position-wise feed-forward network (FFN). These models are designed for understanding tasks, as the effectively capturing contextual dependencies and semantic relationships within the input tokens, however they are not suited for text generation tasks. These abilities provide them with a natural advantage in tasks like text classification and information extraction. Research has also reported that such models demonstrate effectiveness in differentiating between machine-generated and human-written text, thus reducing the risk of misinformation and false content propagation. The most representative model among them is BERT (bidirectional encoder representations from transformers), it introduced masked language modeling (MLM) to enable bidirectional training [3]. Fig. 1 shows the model architecture of BERT. Its variants, such as RoBERTa, ALBERT and others, extend BERT's framework with optimized training objectives.

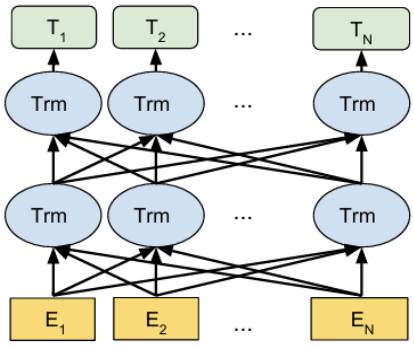

Fig. 1 BERT Model Architecture [3]

2.2. Encoder-decoder Architecture

The fundamental structure in modern transformer-based models. Originally introduced by Vaswani et al., both encoder and decoder consist in the same model [1]. Where the encoder first

converts the input sequence into contextual representations that capture semantic relationships across all tokens. And the decoder then generates the output sequence autoregressively with cross-attention processing. Fig. 2 shows the most basic architecture of the transformer model. This architecture is powerful in tasks like natural language understanding and text generation, it has shown great strength across various natural language processing (NLP) tasks including translation, summarization, and generation of text or code.

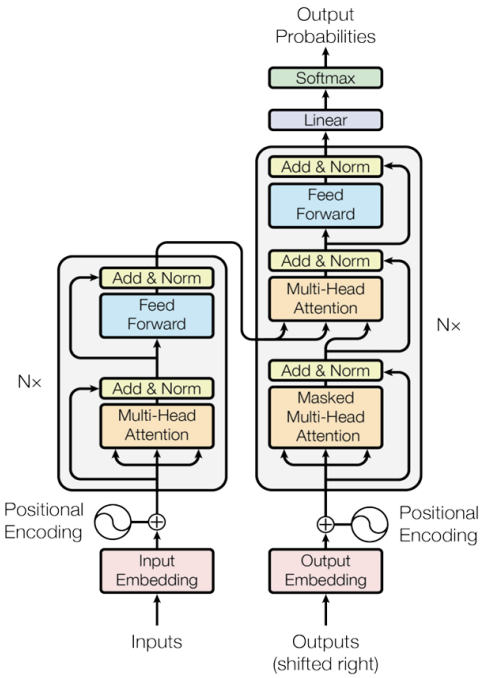

Fig. 2 Encoder-Decoder Transformer Architecture [1]

Well known models include Bidirectional and Auto-Regressive Transformers (BART) and Text-to-Text Transfer Transformer (T5) [4, 5]. BART is a standard encoder-decoder model that has strong performance in summarization and denoising tasks and T5 is famous for reframing all NLP tasks as text-to-text problems.

2.3. Decoder-only Architecture

The most widely adopted structure for modern LLMs. Examples include ChatGPT families, LLaMA, Gemini, and such [6-8]. These models are built upon causal decoders, in which each output token attends only to its previous tokens and itself, Fig. 3 shows the model architecture.

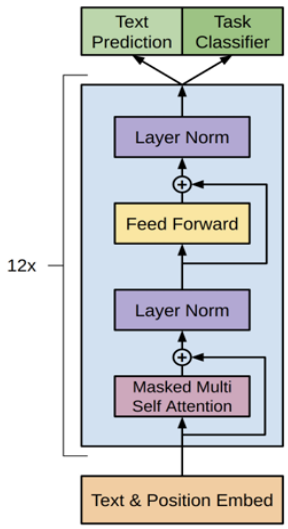

Fig. 3 Decoder-Only Transformer Architecture [6]

A major strength of decoder-only models is that they strongly benefit from the scaling law. As the scaling law suggests, the performance of the models tends to scale predictably with the number of computational resources and training data provided [9]. As the number of parameters continues to grow, often ranging from hundreds of billions to trillions, they are capable of generating more accurate and contextually rich output, solving increasingly complex problems. This phenomenon is commonly referred to as emergence. Despite their success, the models are facing several key challenges. One major limitation is the context length constraint, as the attention scales quadratically with sequence length, leading to high computational and memory costs for long inputs. Hallucination, usually defined as the output content generated is nonsensical or unfaithful, is another challenge that concerns both researchers and users. Not all users have the expertise to verify the correctness of the outputs, misinformation spread to them can undermine the reliability of models' input, and this can be the least severe consequences [10]. In certain domains such as legal practice, healthcare, and autonomous driving, such unreliability may pose serious ethical or life-threatening risks.

2.4. Comparative Study and Limitation Analysis

Different architectures have their own specialties, but when it comes to edge deployment, they do face similar challenges. A major limitation of the edge devices lies in they often cannot host the full-scale model as the memory space is not big enough. And the enormous computational power demanded during inference exceeds devices' available capabilities. Moreover, the inference processes usually have high energy consumption, lots of heat is generated during the process. Most edge environments are not capable to support such consumption and are not able to meet the thermal dissipation requirements. The device will be more likely to shut down due to lack of power or overheating before results are generated. What contributed to models' remarkable performance has now turned into major obstacles for local deployment. Thus, extensive efforts in this area have attempted to address these issues. Model quantization, aiming to reduce the model size to deploy on resource constrained devices through reducing the numerical precision is explored by many works. The complete local deployment effectively preserves privacy and security, but it came at the cost of reduced accuracy. Another stream of research focuses on reducing model parameters, producing what are known as lightweighted models [11]. While sharing the same goal of quantization, these approaches face the same trade-off between model size and accuracy. Some studies have also explored coexistence and collaboration between cloud, edge nodes, and edge devices [12]. This approach partially mitigates privacy risks by processing sensitive data locally, but it also demands high network bandwidth and connection stability.

3. Optimization Analysis

3.1. Quantization

Quantization is one of the most common techniques used to make LLMs more efficient, post-training quantization (PTQ) is one of those several quantization techniques. Shen et al. presented Agile-Quant, an activation-guided quantization strategy that aims to speed up the inference of LLMs on edge device [13]. It introduces the joint quantizes of weights and activations to implement practical acceleration on edge devices. To eliminate the outliers caused by quantization, it is further coupled with activation-aware token pruning method and hardware-aligned kernel design. Experiments were performed on commodity edge CPUs such as the Snapdragon 870, in comparison with previous weight-only quantization approaches. It achieved up to 2.55x speedup across multiple benchmarks and different edge devices. The evaluation focuses mainly on perplexity metrics rather than downstream reasoning or task-specific performance, which leaving its broader applicability uncertain. Nonetheless, the paper reveals the efficiency of the co-design of algorithmic quantization strategies and hardware characteristics for low-bit inference on edge devices without compromising accuracy.

3.2. Mixture of Experts

Mixture of experts (MoE) is a framework that divides a large model into multiple smaller sub-network works called experts, a gated network dynamically selects a small set of experts for different tasks [14].

SwapMoE introduces a framework that serve MoE models on memory-constrained devices [15]. The proposed approach maintains a small subset of Virtual Experts (VE) in main memory, which are dynamically mapped and synchronized with full experts in external storage. It aims to reduce the memory footprint and ensuring low-latency inference without aggressive pruning. Experimental results showed it is able to reduce the memory usage by roughly 60% while halving inference latency and preserving output quality. However, the evaluation focuses mainly on text summarization and language modeling tasks conducted on models of relatively small scale. The limited evaluation does raises uncertainty regarding its scalability to other commonly used tasks and truly large models.

Another study also presents a comprehensive approach to accelerate and optimize MoE model inference on resource-constrained devices [16]. The authors introduce the Fate system, which exploits the similarity of gaiting inputs between adjacent layers to perform cross-layer expert prefetching. By integrating the shallow-favoring expert caching strategy and popularity-aware hybrid quantization strategy, it improves the expert cache hit rates and reduces the I/O stalls. In terms of experimental, the evaluation is conducted on two models, each containing tens of billions of parameters, using PC-class edge GPUs. Results showing a up to 4x speedup over the load-on-demand method and about 2x speedup on the expert activation path-based approach. Similar to the previous study, the evaluation conducted on limited models and setups leave room for optimizations. Fate also assumes a stable cross-layer similarity, which may result in varying performance across different routing and mechanisms. Apart from the optimizable aspects, the sparsely activated MoE approach remains well-suited for edge scenarios, provided that effective strategies for optimizing memory utilization are employed.

3.3. Lightweight Models

Another approach named supernet training is used to generate lightweight submodels from a single, over-parametrized network, the supernet. The goal is to train all possible submodels jointly so that different model configurations (e.g. small, medium, large) can be extracted from one training process. Kundu et al. introduced multistage low-rank fine-tuning of super-transformers (MLFS), a supernet-based approach that enables the generation of a family of submodels through structured weight slicing [17]. It aims to generate the desired model by the slicing operation when performing fine-tuning and cache the resulting subnet for inference. For the encoder-only model families, the experimental results show that submodels distilled archive comparable accuracy to baselines. However, the authors observe the decoder-only model families experience a performance degradation when the subnets are smaller than roughly two-thirds of original size. Considering the unsatisfactory performance of this approach on decoder-only models and the limited scale of models used for evaluation, it's applicability in the current environment, where decoder-only models dominate, remains to be further assessed. Nonetheless, lightweight models could still remain a promising approach for edge deployment of LLMs.

3.4. Hardware Optimization

Apart from adjustments done to the model itself, some works also explored hardware-level approaches. EdgeLLM introduced a CPU-FPGA heterogeneous accelerator [18]. The designed processing element (PE) supports mixed-precision multiplication, performing FP16*FP16 for MHA and FP16*INT4 for FFN operations. An optimized log-scale sparsity mechanism maintains full PE utilization while reducing effective weight bit-width and memory bandwidth demands. On the software co-design side, unified data format is used to eliminate data reshaping, and asynchronous processing between I/O and compute domain maximizes HBM bandwidth utilization. Experimental results show a 1.91x higher throughput and 7.55x higher energy efficiency than a commercial GPU.

The main limitations lie in generality and scalability. The rotary embeddings require either custom hardware implementation for different models or CPU-based processing. Additionally, while the unified data format and compiler design improve efficiency, they introduce complexity in adaptation across diver model architectures.

Another algorithm-hardware co-design example is AccLLM, aims for enabling efficient long-context inference for LLMs on edge devices [19]. On the algorithm level, it integrates 2:4 semi-structured pruning, Λ-shaped attention, and a W2A8KV4 (2-bit weights, 8-bit activation, and 4-bit KV cache) quantization scheme jointly to reduce computational complexity and memory footprint. To fully exploit the advantages of these algorithms, dedicated hardware architectures have been proposed for better adaptation. Including a reconfigurable computing engine (RCE) designed to switch between matrix-matrix and matrix-vector processing, a selector to accelerate sparse workloads, and efficiently pack DSPs for mixed-precision operations. Experimental results show that it reaches 2x to 8x higher throughput and 2x to 20x higher energy efficiency over multiple platforms running diverse models. While the scalability and performance of this design to larger models, diverse tasks, or other more common hardware platforms remains unverified, the paper nonetheless demonstrates the potential of software-hardware co-design approach.

3.5. Cloud-Edge Collaboration

Collaborative edge computing refers to the integration of resources from edge devices and cloud servers to improve resource utilization and performance optimization. EdgeShard presents a framework through this approach, where the LLM is partitioned into shards and are deployed on multiple edge devices [2]. It formulates the deployment problem as a joint optimization problem of device selection and model partitioning under memory, privacy, and bandwidth constraints. Through dynamic programming for latency and throughput optimization, together with offline profiling and online scheduling, it addresses challenges such as device heterogeneity, low and unstable inter-device bandwidth, and the data dependencies introduced by LLM decoding. Experimental results showing a 50% latency reduction and 2x throughput enhancement with running a full-precision model on a 15 devices network setup. However, it raises a notable privacy concern. In typical edge scenarios, it is rare to have access to such a large number of devices for concurrent collaboration. Furthermore, lack of privacy-preserving mechanisms, trust management frameworks, and incentive mechanisms introduces additional challenges on leveraging nearby devices.

PerLLM is another edge-cloud collaboration framework working towards scheduling and resource allocation optimization to reduce latency and energy consumption [20]. The proposed scheduling framework formulates such collaboration as a multi-objective optimization problem, with energy, latency, bandwidth, and computing power as constraints. The constraint satisfaction upper confidence bound (CS-UCB) extended from traditional UCB framework is introduced to handle the multi-constraint decision making challenge. It models the problem as a combinatorial multi-armed bandit (CMAB) process where each “arm” represents a possible resource allocation strategy. Experimental results showing it yields 1.6x to 2.2x higher throughput and over 50% reduction on energy consumption compared with different models. However the evaluation was performed on rather small model sizes, which may limit the generalizability of the proposal. Additionally, the current implementation assumes homogenous edge devices with identical computational capabilities, whereas practical deployments may often involve heterogeneous hardware configurations that may influence scheduling decisions. Similar to EdgeShard, PerLLM also lacks consideration of privacy constraints. Finally, the framework focuses primarily on optimizing energy and latency, without explicitly addressing accuracy and memory sharing.

4. Conclusion

In this paper, an overview of LLMs and introduce different LLM architectures was first provided. We have discussed that, with the growing demand for LLM applications, a number of challenges have

begun to emerge accordingly. Traditional cloud dependent computations are facing the problems of unstable network quality and constrained bandwidth, these challenges introduce increased inference latency and prevents them to meet the requirements under latency-critical use cases. Users are also concerning their privacy as it need constant network access and data transmission to remote server. All of those challenges are motivating researchers to explore the possibilties of deploying the models locally on edge devices. Current edge devices are not able to satisfy such deployment. Their portable design led to limited computational capability, memory capacity, and power budget. Many approaches are proposed to address these issues, which covers multiple directions, including software algorithm-level techniques such as quantization, lightweight or distilled models, and MoE-based architectures; hardware acceleration optimization; algorithm–hardware co-design approaches; and edge–cloud collaboration such as split inference strategies.

This paper provides a summarization on them and reference a range of studies to offer evidence and analysis for the respective approaches. Although extensive efforts are devoted to the related areas, many challenges still remain, and they leave room for new opportunities. We hope this paper provides useful insights into current research status, difficulties, and potential opportunities in this field.

References

[1] Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need. Advances in neural information processing systems, 2017, 30.

[2] Zhang M, Shen X, Cao J, et al. Edgeshard: Efficient llm inference via collaborative edge computing. IEEE Internet of Things Journal, 2024.

[3] Devlin J, Chang M W, Lee K, et al. Bert: Pre-training of deep bidirectional transformers for language understanding. Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers). 2019: 4171-4186.

[4] Lewis M, Liu Y, Goyal N, et al. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. Proceedings of the 58th annual meeting of the association for computational linguistics. 2020: 7871-7880.

[5] Raffel C, Shazeer N, Roberts A, et al. Exploring the limits of transfer learning with a unified text-to-text transformer. Journal of machine learning research, 2020, 21(140): 1-67.

[6] Radford A, Narasimhan K, Salimans T, et al. Improving language understanding by generative pre-training. OpenAI preprint. 2018.

[7] Touvron H, Lavril T, Izacard G, et al. Llama: Open and efficient foundation language models. arXiv preprint arXiv:2302.13971, 2023.

[8] Team G, Anil R, Borgeaud S, et al. Gemini: a family of highly capable multimodal models. arXiv preprint arXiv:2312.11805, 2023.

[9] Kaplan J, McCandlish S, Henighan T, et al. Scaling laws for neural language models. arXiv preprint arXiv:2001.08361, 2020.

[10] Ji Z, Lee N, Frieske R, et al. Survey of hallucination in natural language generation. ACM computing surveys, 2023, 55(12): 1-38.

[11] Wang C, Yan J, Yue Y, et al. DistilQwen2. 5: Industrial Practices of Training Distilled Open Lightweight Language Models. arXiv preprint arXiv:2504.15027, 2025.

[12] Chen Y, Li R, Zhao Z, et al. NetGPT: A native-AI network architecture beyond provisioning personalized generative services. arXiv preprint arXiv:2307.06148, 2023.

[13] Shen X, Dong P, Lu L, et al. Agile-quant: Activation-guided quantization for faster inference of LLMs on the edge. Proceedings of the AAAI Conference on Artificial Intelligence. 2024, 38(17): 18944-18951.

[14] Cai W, Jiang J, Wang F, et al. A survey on mixture of experts in large language models. IEEE Transactions on Knowledge and Data Engineering, 2025.

[15] Kong R, Li Y, Feng Q, et al. SwapMoE: Serving off-the-shelf MoE-based large language models with tunable memory budget. Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). 2024: 6710-6720.

[16] Fang Z, Hong Z, Huang Y, et al. Fate: Fast Edge Inference of Mixture-of-Experts Models via Cross-Layer Gate. arXiv preprint arXiv:2502.12224, 2025.

[17] Kundu A, Lim Y C F, Chew A, et al. Efficiently distilling LLMs for edge applications. Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 6: Industry Track). 2024: 52-62.

[18] Huang M, Shen A, Li K, et al. Edgellm: A highly efficient cpu-fpga heterogeneous edge accelerator for large language models. IEEE Transactions on Circuits and Systems I: Regular Papers, 2025.

[19] Liang Y, Shi H, Shao H, et al. AccLLM: Accelerating Long-Context LLM Inference Via Algorithm-Hardware Co-Design. arXiv preprint arXiv:2505.03745, 2025.

[20] Yang Z, Yang Y, Zhao C, et al. Perllm: Personalized inference scheduling with edge-cloud collaboration for diverse llm services. arXiv preprint arXiv:2405.14636, 2024.