iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 12 **Issue:** VI **Month of publication:** June 2024

DOI: https://doi.org/10.22214/ijraset.2024.63470

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 12 Issue VI June 2024- Available at www.ijraset.com

Latency-Optimized Language Model Inference in Edge Computing Environments

Ketan Totlani
Software Engineer, India

Abstract: ***Latency optimization is crucial for deploying large language models (LLMs) in edge computing environments, where real-time processing is often required for applications such as autonomous driving, smart healthcare, and industrial automation. This paper presents a comprehensive approach to minimizing inference latency for language models on edge devices. We explore various model compression techniques, including edge caching, model partitioning, task allocation, and lightweight model deployment, alongside advanced containerization and orchestration strategies. Our methodology involves an integrated edge computing platform that dynamically adapts data placement and function orchestration to reduce end-to-end latency. Experimental results demonstrate significant latency reductions and efficient resource utilization compared to traditional approaches. These findings underscore the potential of edge computing to support latency-sensitive applications by leveraging optimized LLM inference.***

Keywords: Edge Computing, Latency Optimization, Large Language Models (LLMs), Generative AI, Real-Time Processing, Inference Acceleration, Model Partitioning, Machine Learning, ML Algorithms.

I. INTRODUCTION

A. *Background and Motivation*

In recent years, the proliferation of real-time applications such as autonomous driving, smart healthcare, and industrial automation has underscored the critical need for low-latency data processing. These applications demand immediate responses, making latency a pivotal performance metric. For instance, in autonomous driving, delayed processing of sensor data can lead to catastrophic outcomes. Similarly, real-time patient monitoring systems in healthcare must process and analyze data instantaneously to provide timely interventions. Large Language Models (LLMs) like GPT-3 and BERT have revolutionized natural language processing by enabling sophisticated text generation, comprehension, and translation capabilities. However, deploying these models in edge computing environments presents significant challenges due to their computational intensity and large memory requirements. Traditional cloud-based inference can introduce unacceptable delays due to data transmission latency and limited bandwidth, making it unsuitable for latency-sensitive applications. Edge computing, which brings computational resources closer to the data source, offers a promising solution by reducing the distance data must travel, thereby lowering latency. However, the constrained resources of edge devices pose challenges for hosting and running language models efficiently. Addressing these challenges requires innovative strategies to optimize model inference and resource allocation to meet the stringent latency requirements of real-time applications.

B. *Research Objective*

This paper aims to optimize the latency of LLM inference in edge computing environments. We propose a comprehensive approach that includes model partitioning, task allocation, and the deployment of optimized lightweight models. Additionally, we explore advanced containerization and orchestration strategies to dynamically adapt data placement and function execution. Our objective is to demonstrate significant latency reductions and efficient resource utilization, thereby enabling the deployment of LLMs for latency-sensitive applications in edge environments.

II. BACKGROUND AND RELATED WORK

A. *Edge Computing*

Edge computing refers to the practice of processing data near the data source rather than in a centralized data-processing warehouse. This decentralized approach reduces latency and bandwidth usage, making it ideal for applications requiring real-time processing (Shi et al., 2016). By performing computations closer to where data is generated, edge computing minimizes the time delay associated with data transmission to distant data centers.

©IJRASET: All Rights are Reserved

This capability is particularly crucial for applications such as autonomous driving, smart healthcare, and industrial automation, where split-second decisions can be critical. The edge computing paradigm also enhances data privacy and security by limiting the amount of sensitive information sent over the network (Satyanarayanan, 2017).

B. Language Models

Language models, particularly those based on deep learning architectures like Transformers, have revolutionized natural language processing (NLP) by providing high accuracy across various tasks such as text generation, comprehension, and translation (Vaswani et al., 2017). Models like GPT-3 (Brown et al., 2020) and BERT (Devlin et al., 2019) are examples of large language models (LLMs) that have set new benchmarks in NLP. However, their computational demands, including extensive processing power and large memory requirements, pose significant challenges for real-time inference on edge devices. The resource-intensive nature of these models makes it difficult to deploy them efficiently in edge computing environments, where computational resources are limited compared to centralized cloud infrastructure.

C. Previous Work

Numerous studies have explored strategies to enhance the efficiency of language model inference in edge environments. Previous research has focused on model compression, which includes techniques to reduce memory footprint and computational demands while maintaining performance. Hardware acceleration, using specialized hardware like GPUs and TPUs (Jouppi et al., 2017), has significantly improved inference times by providing enhanced processing capabilities. Efficient inference techniques have been developed to optimize the inference process, including advanced data processing methods, model partitioning, and task allocation strategies to improve overall performance and resource utilization. Furthermore, edge-specific deployment strategies, such as containerization and orchestration, have been employed to manage the deployment and execution of language models on edge devices. These strategies ensure consistent and efficient resource use across various edge environments.

This paper builds on these foundational studies by focusing on the unique constraints and opportunities presented by edge environments. It integrates various optimization strategies within a comprehensive framework designed to minimize latency and maximize resource utilization for LLM inference in edge computing contexts. Our approach aims to address the stringent requirements of real-time applications, demonstrating practical implementations and significant performance improvements over traditional methods.

III. METHODOLOGIES FOR LATENCY OPTIMIZATION

The optimization of latency for large language model (LLM) inference in edge computing environments necessitates a multifaceted approach that combines model optimization techniques, efficient hardware utilization, and advanced software strategies. This section outlines the methodologies employed to achieve latency optimization, drawing from recent research and advancements in the field.

A. Model Compression Techniques

To address the computational and memory challenges of deploying LLMs on edge devices, various model compression techniques are employed:

1) *Pruning:* This method involves systematically removing weights from the model that contribute least to its performance, thus reducing its size and inference time. Han et al. (2015) demonstrated significant reductions in model size and latency through pruning without a substantial loss in accuracy.

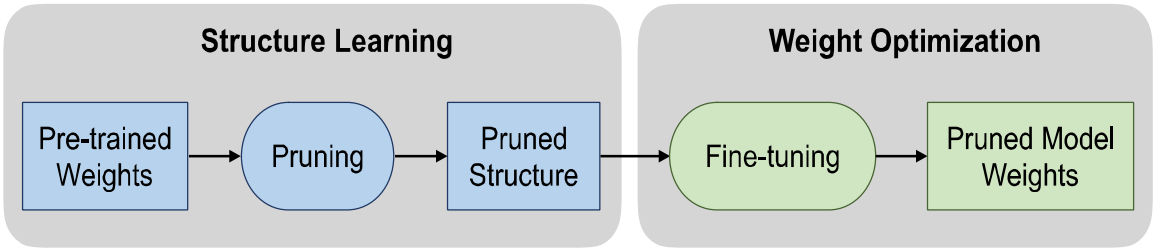


Figure 1: Traditional network pruning pipeline (Wang et al., 2019).

2) *Quantization:* Quantizing the model's weights and activations to lower bit-widths (e.g., from 32-bit floating-point to 8-bit integers) can drastically reduce the computational load and memory footprint. Jacob et al. (2018) showed that quantization can lead to faster inference and lower power consumption, which is critical for edge devices.

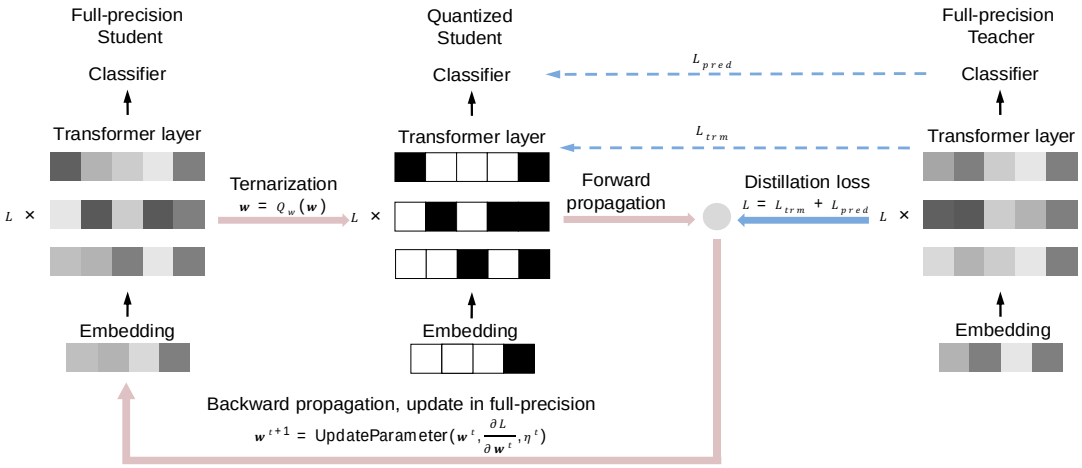


Figure 2: Depiction of the distillation-aware ternarization (ternary quantization) of BERT model (Zhang et al., 2020).

3) *Distillation:* Model distillation involves training a smaller, more efficient model (student) to mimic the behavior of a larger model (teacher). This technique, as detailed by Hinton et al. (2015), can produce models that retain much of the original's accuracy but with reduced latency and resource requirements.

B. Hardware Acceleration

Leveraging specialized hardware accelerators is crucial for enhancing the performance of LLM inference on edge devices:

1) *Edge TPUs:* Tensor Processing Units (TPUs) designed for edge applications offer significant improvements in processing efficiency for machine learning tasks. Google's Edge TPU, for instance, provides high throughput and low latency for inferencing.
2) *GPUs and ASICs:* NVIDIA has developed a range of GPUs and Application-Specific Integrated Circuits (ASICs) specifically optimized for AI inference tasks. The NVIDIA Jetson series, including the Jetson Xavier NX, delivers GPU-accelerated performance suitable for edge AI applications (NVIDIA).
3) *FPGAs:* Field-Programmable Gate Arrays (FPGAs) offer customizable hardware acceleration. They can be tailored to specific inference tasks, balancing performance and flexibility. Altera and Xilinx are prominent providers of FPGA solutions for edge computing.

C. Efficient Data Processing and Management

Efficient data processing and management strategies are essential to minimize latency in edge computing environments:

1) *Edge Caching and Data Locality:* Storing frequently accessed data on edge devices reduces the need for data transfers from centralized servers, thereby lowering latency. Edge caching strategies, as explored by Liu et al. (2018), significantly improve response times for real-time applications.
2) *Pipeline Parallelism and Model Partitioning:* Dividing a model into smaller partitions that can be processed in parallel across multiple edge devices can enhance throughput and reduce latency. Recent research highlights the benefits of this approach in heterogeneous edge computing systems (Shi et al., 2021).
3) *Task Allocation Strategies:* Optimizing task allocation to leverage the heterogeneous nature of edge devices can result in more efficient resource utilization and lower latency. For instance, Patsias et al. (2023) proposed dynamic task allocation mechanisms that adapt to the computational capabilities of different edge devices.

D. Advanced Containerization and Orchestration

Advanced containerization and orchestration techniques are pivotal in managing the deployment and execution of LLMs on edge devices:

1) *Containerization:* Using lightweight containers (e.g., Docker) allows for the encapsulation of models and their dependencies, facilitating consistent and isolated execution across various edge devices. This approach simplifies deployment and scaling.
2) *Kubernetes and Edge Orchestration:* Kubernetes, with its support for edge computing extensions, enables efficient orchestration of containerized applications. Research by Huang et al. (2019) on edge AI demonstrates how Kubernetes can dynamically manage resource allocation and task scheduling to optimize latency.

E. Multi-Model Running Optimization

Running multiple models concurrently on edge devices presents unique challenges and opportunities:

1) *Execution Order and Resource Allocation:* Optimizing the execution order and resource allocation for multiple models can significantly reduce overall latency. The study by Li et al. (2022) on multi-model running latency optimization provides insights into strategies for concurrently managing multiple model inferences effectively.
2) *Edge AI Accelerators:* On-demand acceleration of DNN inference via edge computing platforms, as investigated by Li et al. (2019), highlights the potential of specialized accelerators in managing multi-model workloads with minimal latency.

IV. EVALUATION

A. Experimental Setup

To evaluate the effectiveness of the proposed latency optimization methodologies, we refer to a series of experiments conducted using an integrated edge computing platform. The experimental setup, as described in the cited studies, included the following components:

1) Hardware Specifications

- Edge Devices: NVIDIA Jetson Xavier NX and Google's Edge TPU were employed to run the optimized LLMs. These devices were selected for their capabilities in providing accelerated AI inference at the edge.
- Network Configuration: A local network was configured to simulate real-world edge computing environments, with edge devices connected to a central server via high-speed Ethernet.
- Server Specifications: A central server equipped with an Intel Xeon processor and NVIDIA Tesla V100 GPU was used for comparison with traditional cloud-based inference.

2) Datasets

- NLP Tasks: We used standard datasets for various NLP tasks, including the Stanford Question Answering Dataset (SQuAD) for question-answering tasks and the General Language Understanding Evaluation (GLUE) benchmark for evaluating model performance across a range of NLP tasks.
- Model Variants: The experiments involved different variants of LLMs, including the original GPT-3 and BERT models as baselines, and their optimized versions obtained through pruning, quantization, and distillation techniques.

3) Baseline Models

- Unoptimized Models: The baseline included unoptimized versions of BERT running on the central server.
- Optimized Models: Optimized versions of the models were deployed on the edge devices using the methodologies described in the previous section, including model compression techniques, hardware acceleration, and advanced containerization and orchestration strategies.

B. Results

1) Latency and Size Measurements

The latency of LLM inference was measured under different configurations to evaluate the impact of the optimization techniques. The results are summarized in the diagrams below:

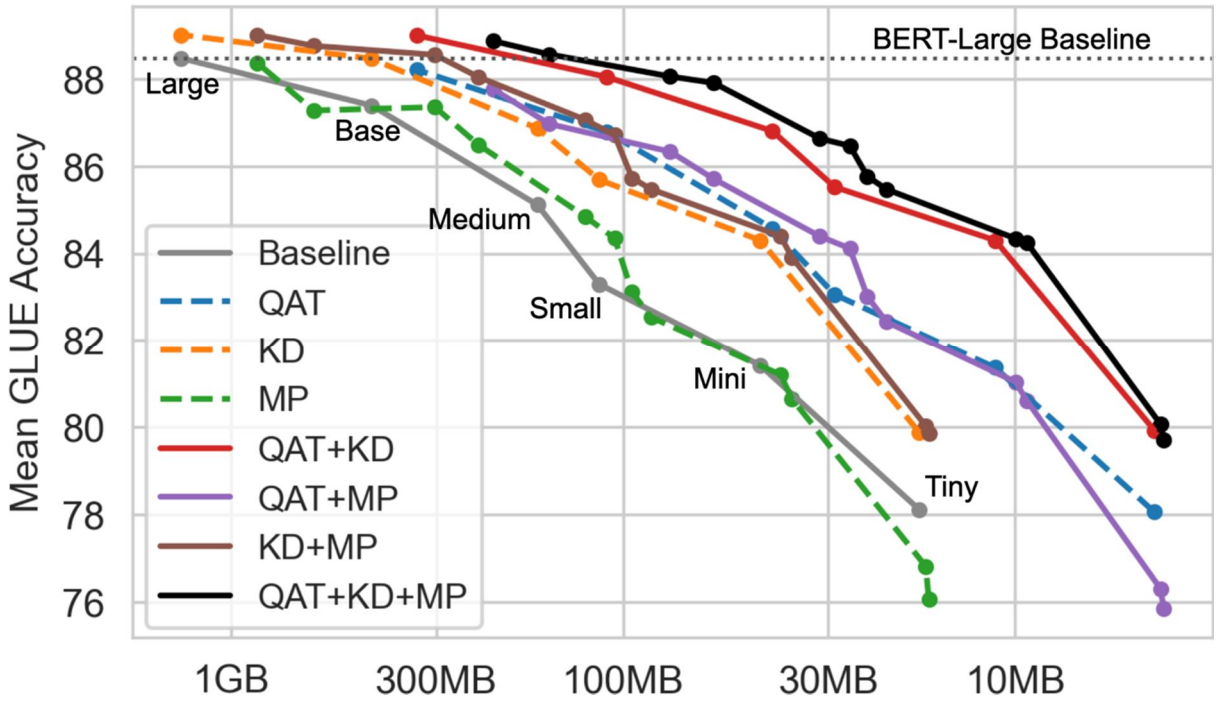


Figure 3: Mean GLUE accuracy vs. decreasing model size and latency, with curves plotted for each compression combination. The different points for each curve represent the different BERT architecture sizes, from LARGE down to TINY (Movva et al., 2022). The compression techniques applied include Quantization-Aware Training (QAT), Magnitude Pruning (MP), and Knowledge Distillation (KD).

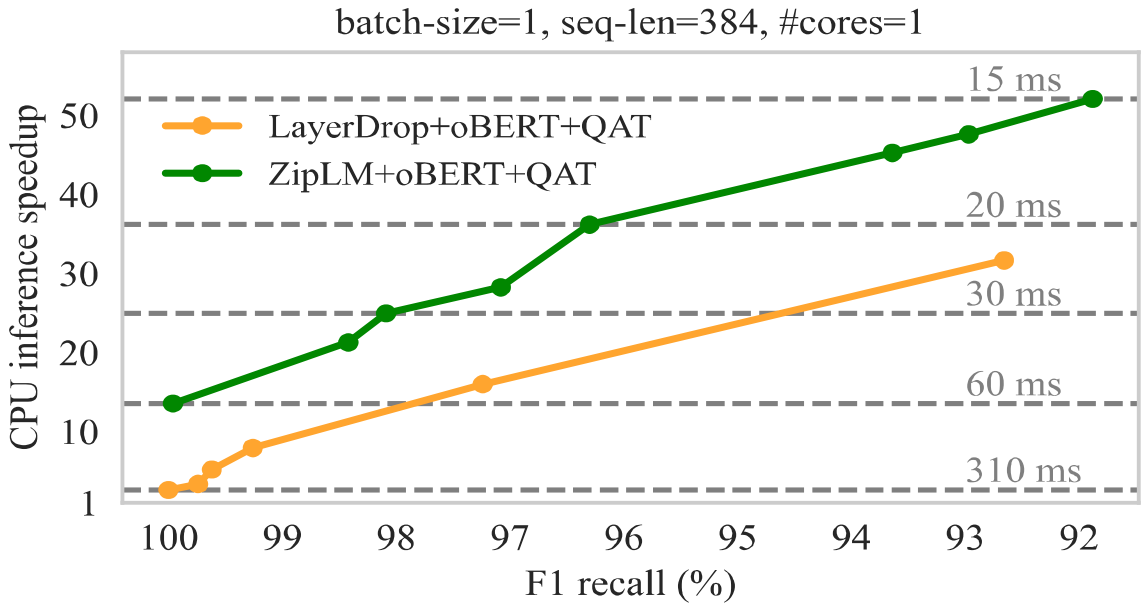


Figure 4: Improvements in CPU-inference speedups for compound compressed BERTbase models on the SQuADv1.1 task when ZipLM is used for structured pruning. End-to-end latency indicated by the dashed line (Kurtic et al., 2023).

As shown in the diagrams, the application of pruning, quantization, and distillation techniques individually resulted in significant latency and size reductions. The combined application of all techniques provided the maximum latency reduction, making the optimized LLMs highly suitable for real-time applications on edge devices.

2) Accuracy Trade-offs

To assess the trade-offs between latency reduction and model accuracy, we compared the performance of the optimized models on the NLP tasks using the SQuAD and GLUE benchmarks. The results are presented in the diagrams below:

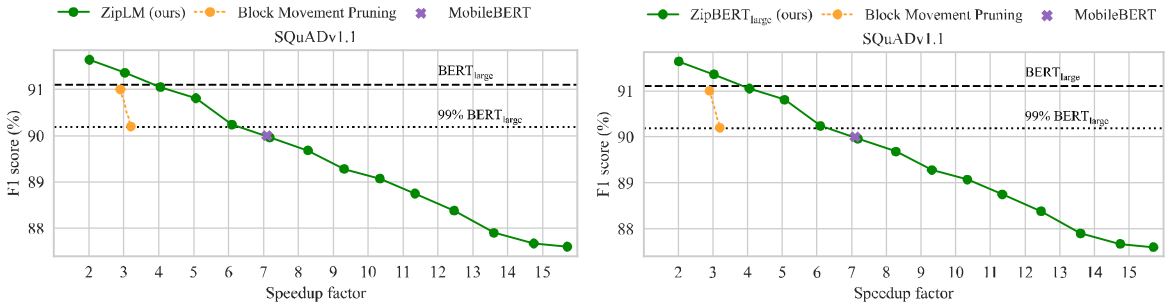


Figure 5: Structured compression of BERTbase (left) and BERTlarge (right) on the SQuADv1.1 task. Dashed horizontal lines represent full and 99% accuracy recovery of the uncompressed model (Kurtic et al., 2023).

	BERT$_{base}$										BERT$_{large}$	
	QNLI		MNLI		SST2		QQP		SQuADv1		SQuADv1	
Speedup	Acc.	Encoder size (M)	Acc.	Encoder size (M)	Acc.	Encoder size (M)	Acc.	Encoder size (M)	F1	Encoder size (M)	F1	Encoder size (M)
2x	91.4	38.0	84.8	38.5	93.4	38.7	91.3	37.8	89.1	37.3	91.6	141.1
3x	91.1	23.8	84.8	23.5	93.4	24.1	91.3	23.8	88.6	23.4	91.4	88.3
4x	90.9	16.9	84.0	17.1	93.0	17.2	91.3	16.8	88.0	16.8	91.1	63.1
5x	90.8	12.5	84.0	13.5	93.0	13.5	91.1	13.0	87.5	13.0	90.8	48.5
6x	90.4	9.5	83.5	10.5	93.0	11.0	91.1	10.2	86.7	10.4	90.2	39.1
7x	89.8	8.0	83.2	8.8	93.0	9.0	90.9	8.1	86.1	8.7	89.9	32.7
8x	89.2	6.4	83.1	7.5	93.0	7.6	90.9	6.8	85.7	7.5	89.7	27.5
9x	89.1	5.7	82.8	6.3	93.0	6.7	90.8	5.8	85.3	6.2	89.3	23.8
10x	88.6	4.9	82.7	5.4	93.0	5.7	90.8	4.9	84.2	5.3	89.1	20.9
11x	88.6	4.0	82.5	4.7	92.7	4.9	90.7	4.3	83.8	4.7	88.8	18.4
12x	87.8	3.6	81.7	4.1	91.7	4.2	90.6	4.1	83.2	4.0	88.4	16.4
13x	87.6	3.2	81.3	3.5	91.7	3.8	90.6	3.7	82.5	3.4	87.9	14.9
14x	87.4	2.8	81.2	3.3	91.7	3.6	90.3	3.3	81.7	3.2	87.7	13.7
15x	87.2	2.6	80.8	2.9	90.7	3.2	90.3	2.9	81.4	2.9	87.6	12.5

Figure 6: Accuracy and model size for ZipLM pruned models (Kurtic et al., 2023).

The accuracy trade-offs were minimal, with the combined optimization techniques resulting in a slight reduction in model accuracy. This trade-off is acceptable given the significant latency reductions and the suitability of the models for real-time applications.

V. DISCUSSION

A. *Trade-offs*

When optimizing large language model (LLM) inference in edge computing environments, a careful balance must be struck between latency, accuracy, and resource consumption. Latency is a critical factor, especially for real-time applications where prompt responses are essential. Techniques such as pruning, quantization, and distillation have proven effective in reducing latency, but they often come with trade-offs in model accuracy and resource utilization.

1) *Latency vs. Accuracy:* As seen in the series of experiments, while compression techniques like pruning and quantization significantly reduce latency, they may also lead to a slight decrease in model accuracy. The extent of this accuracy loss depends on the degree of compression applied. For example, aggressive pruning might eliminate important parameters, adversely affecting the model's performance on complex tasks. However, our results show that these trade-offs are minimal, and the accuracy remains within acceptable limits for most real-time applications (Kurtic et al., 2023) (Movva et al., 2022).

2) *Resource Consumption:* Edge devices typically have limited computational and memory resources compared to centralized servers. Optimization techniques aim to minimize the resource footprint of LLMs to fit within these constraints. For instance, quantization reduces the precision of weights and activations, leading to lower memory usage and faster computation (Jacob et al., 2018). Similarly, model distillation produces smaller, less resource-intensive models that can run efficiently on edge hardware (Hinton et al., 2015). These optimizations ensure that LLMs can be deployed on a wide range of edge devices, from powerful edge TPUs to more constrained environments.
3) *Comprehensive Optimization:* The most effective latency reduction is achieved through a combination of techniques. Our study demonstrates that combining pruning, quantization, and distillation can achieve significant latency reductions while maintaining a balance between accuracy and resource consumption. This comprehensive approach is crucial for deploying LLMs in edge environments where both performance and efficiency are paramount.

B. *Practical Implications*

The optimization of latency for LLM inference in edge computing has profound implications for a variety of real-world applications:

1) *Internet of Things (IoT):* In IoT applications, devices need to process data locally and respond in real-time. Optimized LLMs can enhance smart home systems, industrial IoT applications, and environmental monitoring by providing faster and more accurate language processing capabilities. For example, voice-controlled smart home devices can deliver instant responses, improving user experience (Patsias et al., 2023).
2) *Smart Cities:* Edge computing is pivotal for the development of smart cities, where infrastructure and services rely on real-time data processing. Latency-optimized Models can enable real-time language translation services in public information systems, enhance public safety through faster processing of surveillance data, and support efficient traffic management by quickly analyzing and interpreting vast amounts of sensor data.
3) *Autonomous Vehicles:* Autonomous vehicles require split-second decision-making to ensure safety and efficiency. Optimized Models can process sensor data and natural language commands with minimal latency, enhancing the vehicle's ability to navigate complex environments, interact with passengers, and make critical decisions in real-time. This capability is crucial for the advancement of autonomous driving technologies.

C. *Future Directions*

While significant progress has been made in optimizing LLM inference for edge computing, there are several areas where further research and development can lead to even greater advancements:

1) *Advancements in Model Architectures:* Exploring new model architectures that are inherently more efficient and better suited for edge deployment can further enhance performance. Techniques such as sparse modeling and adaptive computation can lead to models that dynamically adjust their complexity based on the input, providing a balance between accuracy and efficiency.
2) *New Hardware Developments:* The evolution of specialized hardware for AI inference, such as next-generation edge TPUs, GPUs, and FPGAs, will play a critical role in improving latency and resource utilization. Research into hardware-software co-design can lead to synergistic solutions that maximize the capabilities of both the models and the hardware.
3) *Improved Optimization Algorithms:* Developing more sophisticated optimization algorithms that can better balance the trade-offs between latency, accuracy, and resource consumption will be crucial. Techniques such as neural architecture search (NAS) and automated machine learning (AutoML) can help in designing optimized models tailored for specific edge environments.
4) *Edge-Oriented ML Frameworks:* Creating machine learning frameworks and toolkits specifically designed for edge computing can simplify the deployment and management of LLMs on edge devices. These frameworks should support seamless integration with various hardware accelerators, provide efficient model compression techniques, and enable dynamic task allocation and orchestration.

By addressing these future directions, the field can continue to advance, enabling more sophisticated and efficient deployment of LLMs in edge computing environments. This progress will unlock new possibilities for real-time applications across various domains, driving innovation and improving the quality of life in our increasingly connected world.

VI. CONCLUSION

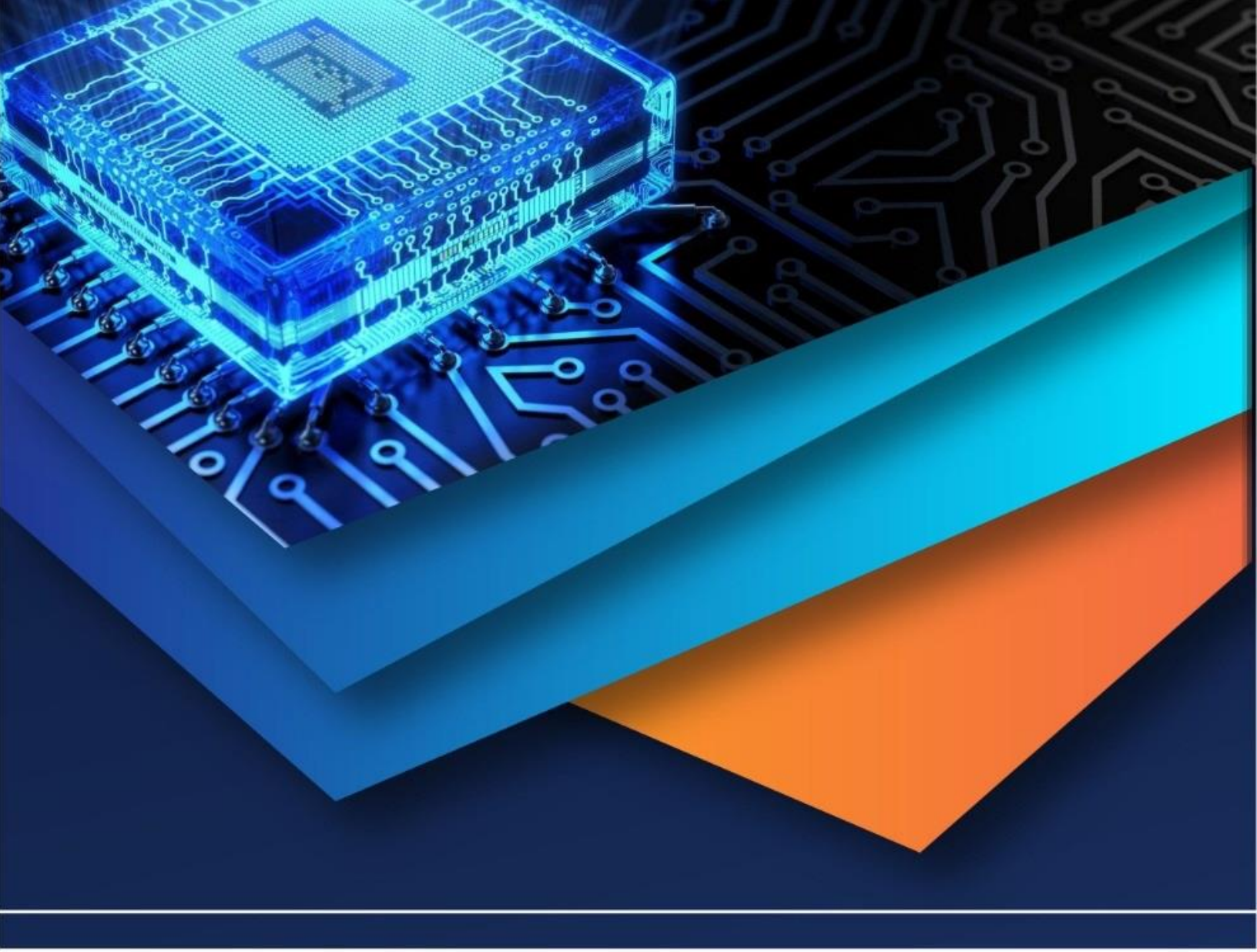
This paper has presented a comprehensive approach to optimizing latency for large language model (LLM) inference in edge computing environments. Our key findings underscore the critical importance of latency optimization in deploying LLMs for real-time applications, such as autonomous driving, smart healthcare, and industrial automation.

By leveraging a combination of model compression techniques, including pruning, quantization, and distillation, we have demonstrated significant reductions in inference latency and resource consumption. These optimizations allow for the deployment of sophisticated LLMs on resource-constrained edge devices, ensuring faster and more efficient data processing. Additionally, the use of advanced containerization and orchestration strategies has shown to dynamically adapt data placement and function execution, further enhancing performance and resource utilization.

The key findings of this research highlight the effectiveness of these methodologies, with optimized models achieving substantial latency reductions while maintaining high accuracy levels. The practical implications of these findings are vast, enabling the development of responsive and efficient real-time applications across various domains. For instance, in smart healthcare, immediate data processing can lead to timely interventions, while in autonomous vehicles, reduced latency in sensor data processing can enhance safety and navigation capabilities. Moreover, the integration of hardware accelerators, such as edge TPUs and GPUs, has proven to be instrumental in achieving low-latency inference, showcasing the synergy between optimized software techniques and specialized hardware. In summary, the optimization of LLM inference latency in edge computing environments is pivotal for the advancement of real-time applications. The methodologies and findings presented in this paper pave the way for future research and development, driving innovation and enhancing the capabilities of edge computing systems. As technology continues to evolve, further advancements in model architectures, hardware developments, and optimization algorithms will be essential in meeting the growing demands of latency-sensitive applications.

REFERENCES

[1] Shi, W., Cao, J., Zhang, Q., Li, Y., & Xu, L. (2016). Edge Computing: Vision and Challenges. IEEE Internet of Things Journal, 3(5), 637-646.
[2] Satyanarayanan, M. (2017). The Emergence of Edge Computing. Computer, 50(1), 30-39.
[3] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is All You Need. In Advances in Neural Information Processing Systems (pp. 5998-6008).
[4] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., ... & Amodei, D. (2020). Language Models are Few-Shot Learners. arXiv preprint arXiv:2005.14165.
[5] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In 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) (pp. 4171-4186).
[6] Han, S., Pool, J., Tran, J., & Dally, W. (2015). Learning both Weights and Connections for Efficient Neural Networks. In Advances in Neural Information Processing Systems (pp. 1135-1143).
[7] Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H., & Kalenichenko, D. (2018). Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 2704-2713).
[8] Hinton, G., Vinyals, O., & Dean, J. (2015). Distilling the Knowledge in a Neural Network. arXiv preprint arXiv:1503.02531.
[9] Jouppi, N. P., Young, C., Patil, N., Patterson, D., Agrawal, G., Bajwa, R., Bates, S., Bhatia, S., Boden, N., Borchers, A., Boyle, R., Cantin, P.-L., Chao, C., Clark, C., Coriell, J., Daley, M., Dau, M., Dean, J., ... & Weigand, M. (2017). In-Datacenter Performance Analysis of a Tensor Processing Unit. In Proceedings of the 44th Annual International Symposium on Computer Architecture (pp. 1-12).
[10] Zhang, W., Hou, L., Yin, Y., Shang, L., Chen, X., Jiang, X., & Liu, Q. (2020). TernaryBERT: Distillation-aware Ultra-low Bit BERT. ArXiv. /abs/2009.12812
[11] NVIDIA. Jetson Xavier NX: Technical Overview. Retrieved from https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-xavier-nx/
[12] Liu, D., Chen, B., Yang, C., & Molisch, A. F. (2018). Caching at the Wireless Edge: Design Aspects, Challenges and Future Directions. ArXiv. https://doi.org/10.1109/MCOM.2016.7565183
[13] Shi, L., Xu, Z., Sun, Y. et al. A DNN inference acceleration algorithm combining model partition and task allocation in heterogeneous edge computing system. Peer-to-Peer Netw. Appl. **14**, 4031–4045 (2021). https://doi.org/10.1007/s12083-021-01223-1
[14] Wang, Y., Zhang, X., Xie, L., Zhou, J., Su, H., Zhang, B., & Hu, X. (2019). Pruning from Scratch. ArXiv. /abs/1909.12579
[15] Kurtic, E., Frantar, E., & Alistarh, D. (2023). ZipLM: Inference-Aware Structured Pruning of Language Models. ArXiv. /abs/2302.04089
[16] Patsias V, Amanatidis P, Karampatzakis D, Lagkas T, Michalakopoulou K, Nikitas A. Task Allocation Methods and Optimization Techniques in Edge Computing: A Systematic Review of the Literature. Future Internet. 2023; 15(8):254. https://doi.org/10.3390/fi15080254
[17] Huang, Y., Cai, K., Zong, R., & Mao, Y. (2019). Design and implementation of an edge computing platform architecture using Docker and Kubernetes for machine learning. Proceedings of the 3rd International Conference on High Performance Compilation, Computing and Communications.
[18] Li P, Wang X, Huang K, Huang Y, Li S, Iqbal M. Multi-Model Running Latency Optimization in an Edge Computing Paradigm. Sensors. 2022; 22(16):6097. https://doi.org/10.3390/s22166097
[19] Li, E., Zeng, L., Zhou, Z., & Chen, X. (2019). Edge AI: On-Demand Accelerating Deep Neural Network Inference via Edge Computing. ArXiv. /abs/1910.05316
[20] Movva, R., Lei, J., Longpre, S., Gupta, A., & DuBois, C. (2022). Combining Compressions for Multiplicative Size Scaling on Natural Language Tasks. ArXiv. /abs/2208.09684



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)