A Survey of Large Language Models and Hardware Optimization

www.surveyx.cn

Abstract

The interdisciplinary field encompassing Large Language Models (LLMs), hardware optimization, AI agent construction, and computational efficiency is pivotal in advancing AI technologies. This survey explores the integration of LLMs with hardware systems, highlighting their transformative potential in enhancing natural language processing, optimizing hardware performance, and constructing intelligent AI agents. The deployment of LLMs on edge devices and their application in multilingual contexts showcase their versatility and adaptability. Innovative hardware strategies, such as FPGA-based accelerators and near-storage processing, are crucial for optimizing LLM operations, reducing computational costs, and improving inference efficiency. The survey also examines frameworks for AI agent construction, emphasizing adaptability and efficiency in dynamic environments. Furthermore, the role of LLMs in AI hardware integration is explored, underscoring their contribution to enhancing system functionality and security. Challenges such as resource management, scalability, and ethical implications are addressed, with solutions involving quantization techniques, adaptive software frameworks, and efficient memory management. Future directions include expanding linguistic diversity, optimizing translation models, and enhancing instructional strategies in education. Overall, the advancements in LLMs and hardware optimization drive innovation across sectors, offering new avenues for research and application in AI systems.

1 Introduction

1.1 Significance of Large Language Models in AI

Large Language Models (LLMs) play a critical role in the advancement of artificial intelligence, demonstrating transformative potential across various domains. In cybersecurity, LLMs enhance honeypot systems, thereby improving data collection methodologies [1]. Their ability to extract structured data from unstructured text via SQL queries facilitates a hybrid querying approach, integrating AI-driven insights into traditional databases [2].

Deploying LLMs on resource-constrained edge devices presents operational challenges but also extends AI capabilities to decentralized environments [3]. In natural language processing, LLMs excel, particularly in machine translation, showcasing their versatility across tasks [4]. The impact of LLMs also reaches computing education, where they reshape pedagogical practices and enhance learning experiences [5].

Dynamic learning approaches have significantly improved multilingual LLM performance, achieving enhancements over baseline methods, which is crucial for global applications [6]. In specialized fields like dentistry, LLMs such as ChatGPT offer promise for advancing automated cross-modal diagnostic methods, highlighting the need for innovative medical AI applications [7].

The development of AI agents capable of mastering complex interactions, such as in the game of Diplomacy, exemplifies LLMs' role in pushing AI boundaries [8]. As LLMs evolve, they con-

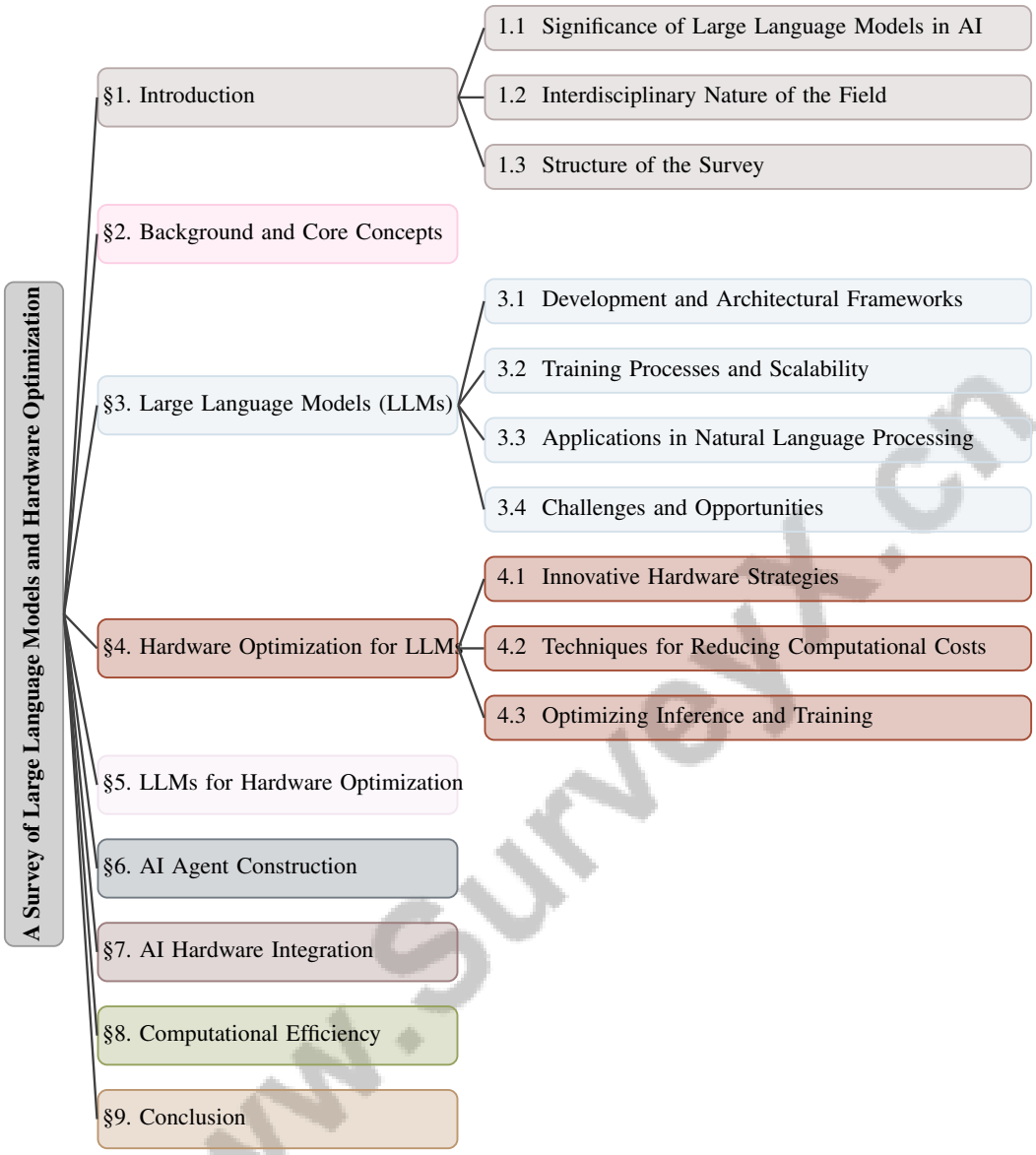


Figure 1: chapter structure

tinue to drive advancements in decision-making, data processing, and domain-specific applications, underscoring their significance in the ongoing evolution of AI technologies.

1.2 Interdisciplinary Nature of the Field

The interdisciplinary nature of the field involving LLMs, hardware optimization, and AI agent construction reflects collaborative efforts across domains to drive innovation and tackle complex challenges. The integration of LLMs with hardware optimization techniques, as illustrated by the EdgeLLM framework, employs CPU-FPGA heterogeneous acceleration to enhance LLM execution on edge devices [3]. This approach not only optimizes performance but also necessitates customized hardware operators and universal data parallelism schemes.

In industrial control systems, the intersection of LLMs with specialized programming environments, such as the Agents4PLC framework, automates closed-loop control processes [9]. Similarly, in financial forecasting, dynamic adaptive optimization techniques merge LLMs, hardware optimization, and AI agent construction to improve predictive accuracy [10].

The integration of architecture, efficient vision, efficient LLMs, training methodologies, data, benchmarks, and applications highlights the collaborative nature of the field, facilitating the development of efficient multimodal LLMs [11]. Moreover, the intersection of LLMs with ontology learning advances knowledge representation and reasoning by automating knowledge extraction from unstructured text [12].

In autonomous driving, the development of multimodal LLMs enhances decision-making processes in vehicles, improving natural language interactions and contextual understanding, thereby showcasing the synergy between AI and transportation technologies [13]. Additionally, the application of LLMs across diverse domains such as education, healthcare, reasoning, text generation, and scientific research illustrates collaborative efforts to address domain-specific challenges [14]. The development of reliable alternatives to manage LLM behaviors, essential for AI governance, further emphasizes interdisciplinary efforts to ensure safe AI systems [15].

The generation of VHDL code for RISC components using LLMs exemplifies the intersection of AI with hardware design, automating hardware description in digital systems [16]. The AutoFlow framework's automated workflow generation underscores the collaborative nature of LLMs, hardware optimization, and AI agent construction, advancing automated system design and optimization [17].

In computing education, the interdisciplinary nature of LLMs is evident in literature reviews, instructional approaches, and ethical implications, which are crucial for shaping educational practices [5]. The construction of AI agents in strategic planning and negotiation contexts integrates LLMs with memory management to enhance agent capabilities [8].

This field thrives on the collaborative integration of diverse disciplines, fostering innovation and effectively addressing complex challenges. The synergy between LLMs and optimization techniques enhances decision-making in dynamic environments, transforming expert intuition into quantifiable features that improve model performance and risk assessment. The imperative for transparency in LLM development, grounded in human-centered design principles, highlights the need for responsible AI practices that consider diverse stakeholder perspectives. Collectively, these elements illustrate the multifaceted nature of advancements in AI and the critical role of interdisciplinary collaboration in driving progress [18, 19, 20].

1.3 Structure of the Survey

This survey is organized into several key sections, each addressing distinct aspects of large language models (LLMs) and hardware optimization to provide a comprehensive overview of the field. The paper begins with an **Introduction**, establishing the significance of LLMs in AI and highlighting the interdisciplinary nature of the field. The **Background and Core Concepts** section defines essential terms and explores the evolution of LLMs and natural language processing technologies.

The **Large Language Models (LLMs)** section delves into their development, architectural frameworks, training processes, scalability challenges, and applications in natural language processing, along with challenges and opportunities within the field. The **Hardware Optimization for LLMs** section examines strategies and technologies for optimizing hardware for LLMs, including innovative strategies to reduce computational costs and optimize inference and training processes.

In the **LLMs for Hardware Optimization** section, the survey investigates how LLMs enhance hardware design and functionality, discussing AI-driven design processes and optimization techniques. The **AI Agent Construction** section explores the construction of intelligent AI agents using LLMs, focusing on frameworks, methodologies, and integration into AI systems.

The **AI Hardware Integration** section highlights the integration of AI capabilities into hardware systems, emphasizing the role of LLMs and solutions for effective integration. The survey addresses the importance of **Computational Efficiency**, discussing methods for enhancing efficiency, algorithmic optimizations, and hardware-software co-design.

Finally, the **Conclusion** summarizes key findings, reflects on current trends, and outlines future directions in the field. This structured approach facilitates a comprehensive examination of the intricate relationship between LLMs and hardware optimization, yielding crucial insights into the computational demands and architectural requirements necessary for the effective training and deployment of advanced AI technologies. By analyzing the interplay of algorithm design, hardware infrastructure, and software optimization, this exploration highlights the need for heterogeneous

computing systems and advanced memory management strategies to meet the escalating performance demands of next-generation LLMs, ultimately guiding the development of more efficient AI computing platforms [19, 21, 22].The following sections are organized as shown in Figure 1.

2 Background and Core Concepts

2.1 Defining Key Terms and Concepts

Large language models (LLMs) are advanced AI systems pivotal in natural language processing (NLP), enabling sophisticated text generation and comprehension [19]. They underpin hybrid querying methods by integrating LLM capabilities with traditional database querying, enhancing data retrieval and analysis [7]. Key to optimizing LLM performance are 'KV caches', which store intermediate results during multi-turn conversations, reducing redundant computations and ensuring real-time responsiveness [23]. Addressing performance gaps in multilingual processing, particularly between English and low-resource languages, necessitates dynamic learning strategies due to limited multilingual training data [24].

AI agents, leveraging LLMs for strategic planning and decision-making, are deployed across diverse domains, including strategic games and autonomous systems, demonstrating their adaptability in complex environments [8]. In healthcare, LLMs enhance diagnostic methods and patient interactions, showcasing AI's transformative potential in medical practices [7]. Optimization algorithms are crucial for improving computational efficiency and hardware utilization, meeting the substantial processing demands of LLMs [19].

These foundational concepts elucidate the relationship between LLMs and hardware optimization, emphasizing the need for advanced solutions like GPUs and TPUs to manage their computational demands. LLMs' applications, from machine translation to real-time data analysis, reveal their innovative potential while highlighting challenges posed by their complexity. Ongoing research into optimized computing architectures and distributed systems is essential for their broader deployment in the AI ecosystem [25, 19, 21, 22].

2.2 The Evolution of LLMs and NLP

The evolution of large language models (LLMs) and natural language processing (NLP) technologies has been driven by significant advancements in model architectures, training methodologies, and application domains, facilitated by increased data availability and computational power [26]. The transition from traditional pre-trained models to advanced multilingual LLMs has broadened linguistic coverage and improved performance across languages [11]. However, inefficiencies in tokenization, especially in non-English languages, pose challenges, necessitating more effective techniques to enhance linguistic capabilities [27]. Integrating knowledge graphs into LLMs has enriched their reasoning capabilities by incorporating structured knowledge [12].

LLMs face challenges such as high training costs and catastrophic forgetting in continual pre-training, prompting innovative solutions [13]. Techniques like rational metareasoning improve computational efficiency, reducing token generation significantly [15]. New benchmarks for natural language to task language (NL-TL) translation aim for flexibility and generalizability, addressing limitations of domain-specific datasets [28]. In multi-objective optimization, LLMs enhance decision-making by inferring optimized solutions where traditional methods falter [29]. Data augmentation (DA) techniques, categorized into generative and discriminative approaches, further enhance LLM training by serving as data annotators and generators [11].

The evolution of LLMs and NLP technologies reflects a rapidly advancing landscape marked by innovations and challenges. Progress in model architecture, training methodologies, and application domains underscores the potential of LLMs in real-world scenarios, such as table-to-text generation and automating systematic literature reviews, while addressing ethical considerations like transparency and accountability. As researchers explore LLM integration with optimization algorithms, the necessity for responsible AI practices becomes clear, highlighting the importance of interdisciplinary collaboration to guide their development and application across various fields [30, 25, 31, 18, 19].

3 Large Language Models (LLMs)

Large language models (LLMs) are transforming artificial intelligence by enhancing capabilities across various domains. Understanding their foundational development is crucial for leveraging their potential. As depicted in Figure 2, this figure illustrates the hierarchical structure of LLMs, detailing their development and architectural frameworks, training processes, scalability, applications in natural language processing, and the associated challenges and opportunities. Each primary category is further divided into subcategories, highlighting key methodologies, applications, and innovations within the domain of LLMs. The following subsection explores these essential frameworks that underpin LLMs' functionality and performance in diverse applications.

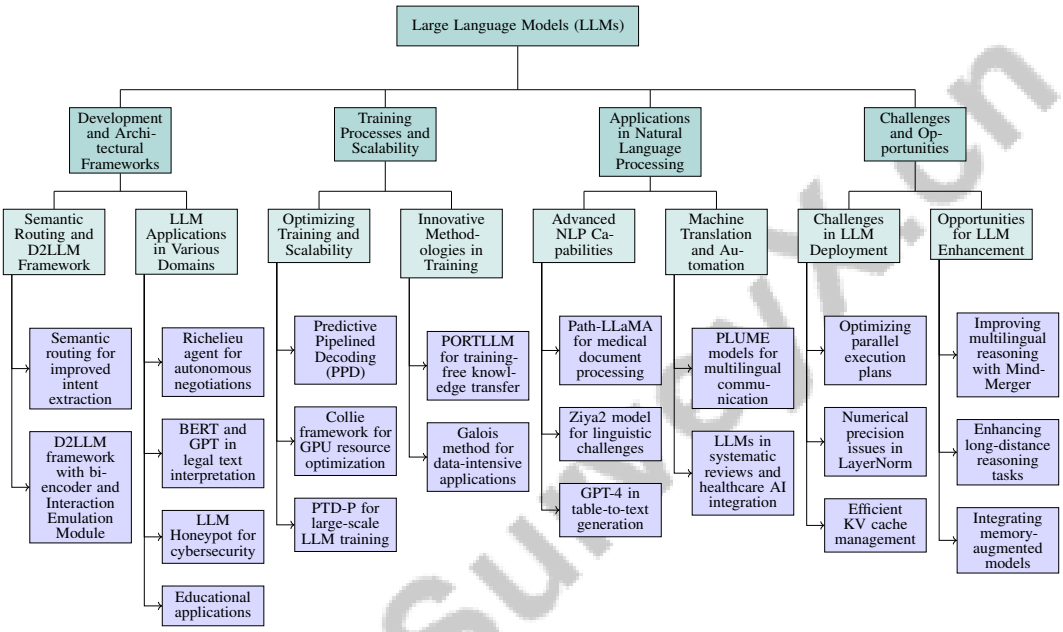


Figure 2: This figure illustrates the hierarchical structure of Large Language Models (LLMs), detailing their development and architectural frameworks, training processes and scalability, applications in natural language processing, and the associated challenges and opportunities. Each primary category is further divided into subcategories, highlighting key methodologies, applications, and innovations within the domain of LLMs.

3.1 Development and Architectural Frameworks

The architectural frameworks of large language models (LLMs) are central to their enhanced performance and functionality across applications. The semantic routing method introduces a deterministic approach to improve intent extraction reliability, addressing existing system limitations [32]. The D2LLM framework, with its decomposed and distilled model integrating a bi-encoder and an Interaction Emulation Module, exemplifies efficient semantic search architecture [27]. Similarly, the Richelieu agent employs LLM-based frameworks for autonomous diplomatic negotiations, highlighting strategic planning integration [8].

In legal text interpretation, BERT and GPT models illustrate advancements in interpreting complex texts, showcasing LLMs' adaptability to specialized domains [33]. The LLM Honeypot framework uses a fine-tuned LLM to simulate a Linux server, enhancing cybersecurity by engaging attackers in a controlled environment [1]. Educational applications also benefit from LLM frameworks, which categorize research into themes like performance assessment and instructional innovation [5].

In autonomous driving, multimodal large language models (MLLMs) improve vehicle intelligence and user interactions, emphasizing the role of architectural frameworks in enhancing decision-making [13]. The MindMerger method bridges performance gaps in multilingual contexts by integrating reasoning and language understanding capabilities through a two-stage training process [24]. LLMs

also advance clinical decision-making by integrating various data sources for automated dental and cross-modal diagnosis [7].

These architectural innovations reflect the dynamic evolution of LLMs, driven by the need for efficiency, adaptability, and application-specific customization. Recent advancements expand LLMs' capabilities, addressing ethical challenges such as transparency and fairness through tailored frameworks and dynamic auditing systems [31, 18, 20].

This evolution is illustrated in Figure 3, which highlights key architectural frameworks in LLMs. The first subfigure contrasts requirements for LLaMA3-70B and GPT-4, emphasizing trade-offs like memory bandwidth and computational power [22]. The second subfigure compares GPT models' access structures, crucial for understanding deployment differences [34]. Finally, the third subfigure presents a GitLab API client in a code editor, demonstrating LLM implementation in applications [35]. Collectively, these examples underscore the significance of Semantic Routing for intent extraction, the D2LLM framework for semantic search, and the Richelieu Agent for strategic planning in diplomatic negotiations.

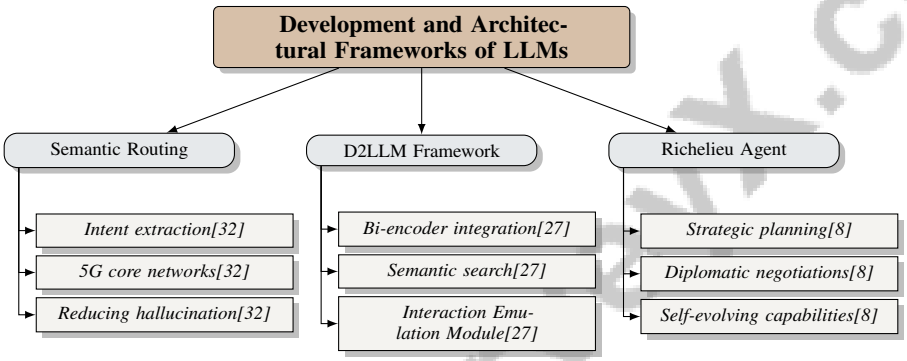


Figure 3: This figure illustrates key architectural frameworks in LLMs, highlighting Semantic Routing for intent extraction, the D2LLM framework for semantic search, and the Richelieu Agent for strategic planning in diplomatic negotiations.

3.2 Training Processes and Scalability

Training large language models (LLMs) involves complex processes and significant computational resources to achieve scalability and efficiency. Predictive Pipelined Decoding (PPD) uses intermediate transformer outputs to predict tokens, reducing decoding latency [36]. The Collie framework optimizes resource utilization across GPUs, improving LLM scalability [37]. PTD-P integrates pipeline, tensor, and data parallelism, enabling large-scale LLM training [38].

The PORTLLM framework introduces a training-free knowledge transfer approach, reducing computational resource burdens [39]. Experiments on tasks like Commonsense Machine Translation and Counter-Intuitive Arithmetic Reasoning highlight LLMs' diverse capabilities [40]. The Galois method integrates LLMs with traditional querying techniques, enhancing scalability in data-intensive applications [2].

Addressing training and scalability challenges requires innovative methodologies and frameworks to optimize resource utilization. Advanced computing systems and heterogeneous architectures are essential for managing LLMs' increasing complexity and resource demands, paving the way for effective solutions in data-constrained environments [30, 41, 42, 25, 21].

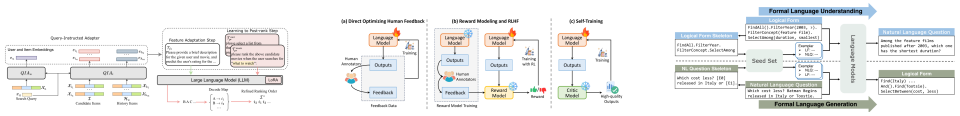
3.3 Applications in Natural Language Processing

Large language models (LLMs) significantly advance natural language processing (NLP) by handling complex tasks and improving computational efficiency. The Path-LLaMA model excels in extracting structured codes from pathology reports, enhancing medical document processing [43]. LLMs also facilitate automated code generation, bridging human language and machine-readable code [44].

The Ziya2 model outperforms baselines across benchmarks, demonstrating versatility in linguistic challenges [45]. In table-to-text generation, GPT-4 creates coherent narratives from structured

data, showcasing advanced capabilities [25]. The D2LLM framework enhances semantic search by combining cross-encoder accuracy with bi-encoder efficiency [27].

In machine translation, PLUME models achieve competitive performance, enhancing multilingual communication [4]. LLMs automate systematic reviews, enhance table-to-text generation, and integrate AI into healthcare, reshaping methodologies across domains [30, 46, 25].



(a) Learning to Post-rank with a Large Language Model (LLM)[47]

(b) Optimizing Language Models with Feedback and Reward Modeling[48]

(c) Formal Language Understanding and Generation[49]

Figure 4: Examples of Applications in Natural Language Processing

In Figure 4, LLMs' transformative impact on NLP is illustrated. The first application enhances post-ranking predictions through a query-instructed adapter [47]. The second application optimizes language models using feedback and reward modeling [48]. The third application details the process of understanding and generating formal language [49].

3.4 Challenges and Opportunities

Deploying large language models (LLMs) involves challenges that impact their effectiveness and scalability. Optimizing parallel execution plans for LLMs is complex due to diverse input characteristics and parallelism trade-offs [50]. High computational resource demands necessitate advanced optimization algorithms for efficient resource management [19].

Numerical precision issues in LayerNorm computations can impair performance. The SLaNC technique improves accuracy and efficiency in LayerNorm computations [51]. Efficient KV cache management, as demonstrated by the CachedAttention mechanism, reduces inference costs and enhances throughput [23].

Multilingual reasoning challenges, particularly in low-resource languages, are addressed by the MindMerger approach, improving accuracy across linguistic contexts [24]. The vast decision space in strategic applications presents opportunities for developing sophisticated LLM-based agents [8].

Opportunities include improving LLM performance in long-distance reasoning tasks through methods like end-to-end graph flattening [52]. Integrating memory-augmented models and instruction tuning enhances summarization and long-term analysis tasks, advancing model performance and versatility.

Addressing LLM challenges and leveraging opportunities is crucial for their development. Innovations like LLM2LLM for data augmentation enhance deployment, expanding applicability across domains [30, 53, 42, 25].

4 Hardware Optimization for LLMs

Enhancing the performance and efficiency of large language models (LLMs) necessitates innovative hardware optimization strategies. Table 1 presents a detailed classification of methods aimed at optimizing hardware for large language models, emphasizing innovative strategies and techniques for enhancing computational efficiency and reducing costs. Additionally, Table 2 presents a comprehensive overview of diverse hardware optimization strategies for large language models, detailing the methods employed to enhance computational efficiency and reduce costs. This section delves into advanced hardware approaches that meet the computational demands of LLMs, highlighting key advancements in hardware design and deployment that support their effective operation across various applications.

Category	Feature	Method
Innovative Hardware Strategies	Proximity-Based Processing	SI[54]
	Dynamic Workflow Adaptation	AF[17]
	Holistic Task Management	RT[55]
	Precision and Efficiency	SLaNC[51]
Techniques for Reducing Computational Costs	Efficiency Techniques	EJM[56], LLM-H[1]
	Hardware Optimization	CLLI[57]
	Integration Strategies	Galois[2]
Optimizing Inference and Training	Efficiency Improvements	APEX[50], MA-LMM[58], FP6[59], ELLI[60], CA[23]

Table 1: This table provides a comprehensive overview of various methods employed to enhance the performance and efficiency of large language models (LLMs) through innovative hardware strategies, techniques for reducing computational costs, and optimization of inference and training processes. Each method is categorized based on its primary focus area, highlighting the diverse approaches adopted to meet the computational demands and improve the deployment of LLMs across different applications.

4.1 Innovative Hardware Strategies

Innovative hardware strategies are pivotal for optimizing LLM operations, which require substantial computational and memory resources. Efficient processing on hardware accelerators often involves overcoming challenges associated with quantization techniques using lower precision formats. The SLaNC method exemplifies this by enabling precise LayerNorm computations, enhancing model performance and inference efficiency [51]. The SECDA-LLM platform optimizes FPGA resources within the llama.cpp inference framework, reducing latency and enhancing performance [19]. Additionally, the TC-FPx framework supports float-point weights across various quantization bit-widths, underscoring the importance of flexible quantization in hardware performance optimization [59].

The Smart-Infinity framework employs near-storage processing devices for parameter updates, minimizing data transfer and significantly boosting training speed and efficiency [54]. Furthermore, the AutoFlow framework enhances LLM adaptability through fine-tuning and in-context methods for workflow generation [17]. Educational contexts benefit from LLMs reducing instructor workloads and improving student experiences, showcasing innovative strategies in educational process optimization [5]. The integration of code execution capabilities within LLMs marks a significant advancement, enhancing operational efficiency [61].

The RoboTool framework exemplifies innovative hardware strategies by integrating components such as Analyzer, Planner, Calculator, and Coder to optimize task execution [55]. Additionally, the fusion of operations within the decoder layer and a segment KV cache policy improve memory management and reduce redundant memory usage [60]. These strategies reflect ongoing efforts to enhance LLM infrastructure, addressing computational challenges and the demand for scalable, efficient systems necessary for deploying advanced AI technologies. As LLMs grow more complex and resource-intensive, optimizing hardware platforms—such as GPUs, TPUs, and FPGAs—becomes critical, aiming to improve computational efficiency and memory management while facilitating effective deployment across diverse applications, ensuring performance requirements are met while managing costs effectively [60, 22, 3, 21, 62].

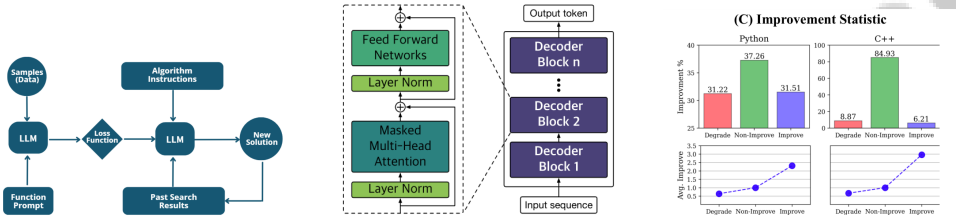
4.2 Techniques for Reducing Computational Costs

Minimizing computational expenses in LLM deployment is essential for enhancing scalability and efficiency. A significant challenge is the underutilization of GPU resources for small-batch inputs, leading to high communication overhead and inefficient memory usage during LLM inference [63]. Innovative strategies are required to optimize resource utilization and streamline computational processes.

The Galois system exemplifies an approach combining SQL's expressive power with LLMs' vast knowledge, enabling accurate and flexible data retrieval [2]. This integration highlights designing systems that balance computational load with retrieval accuracy, reducing overall computational expenses. Efficient 6-bit quantization, as demonstrated by the FP6 approach, reduces memory usage and improves inference speed without sacrificing model quality [59]. Techniques like Low-Rank Adaptation (LoRA) and Quantized Low-Rank Adapters (QLoRA) reduce computational expenses during fine-tuning [1].

Deploying LLMs on CPUs poses additional challenges, necessitating efficient methods for CPU deployment, as existing solutions are often optimized for GPU environments [64]. High latency and low throughput during autoregressive token generation present core challenges in LLM inference [60]. Innovative decoding methods and memory management strategies are essential for enhancing inference efficiency and reducing computational costs. CachedAttention proposes a new attention mechanism that saves and reuses KV caches, reducing repetitive computations [23].

Advanced quantization techniques, including any-precision and mixed-precision methods, along with optimized deployment frameworks like UELLM and efficient inference methodologies, establish a robust strategy for significantly reducing computational costs associated with LLM deployment. These approaches enhance resource utilization, minimize latency, and accommodate multiple LLMs of varying sizes and precision, improving overall performance and efficiency in real-time applications [65, 66, 67, 64, 68]. Such advancements are critical for enabling scalable and efficient use of LLMs across diverse domains and applications, ensuring their continued evolution and practical utility.



(a) The image depicts a flowchart illustrating the process of training a machine learning model (LLM) using a loss function and past search results.[69]

(b) Decoder Network Architecture[57]

(c) Improvement Statistic[56]

Figure 5: Examples of Techniques for Reducing Computational Costs

As shown in Figure 5, reducing computational costs in optimizing hardware for LLMs is a critical challenge that researchers actively address. The accompanying figures provide visual insights into several techniques employed to tackle this issue. The first image presents a flowchart outlining the intricate process of training an LLM by leveraging a loss function alongside historical search results. This serves as a foundational guide, illustrating interconnected stages from data sampling to the application of the loss function, essential for fine-tuning model performance. The second image focuses on the decoder network architecture, a sophisticated neural framework designed for token sequence generation, characterized by its layered structure, incorporating feed-forward networks and masked multi-head attention layers, each contributing to efficient processing and output generation. Finally, the third image offers a comparative analysis of improvement statistics between Python and C++ through bar charts and line graphs, highlighting varying degrees of enhancement across different categories. Collectively, these visual examples underscore the multifaceted strategies being explored to optimize hardware efficiency and reduce the computational demands of LLMs, paving the way for more sustainable and scalable AI solutions [69, 57, 56].

4.3 Optimizing Inference and Training

Enhancing inference and training efficiency for LLMs is crucial for effective deployment across diverse applications. Advanced methodologies, such as TC-FPx, enable efficient matrix multiplication for LLMs with 6-bit quantization, addressing memory access and de-quantization challenges, thus optimizing computational efficiency [59].

Innovative solutions like the Latency Processing Unit (LPU) significantly enhance LLM inference efficiency by optimizing memory bandwidth utilization and reducing latency. This is achieved through a carefully designed architecture that balances memory bandwidth and computational logic, along with advanced synchronization techniques that effectively hide data synchronization delays among multiple LPUs. The LPU can achieve impressive inference speeds of 1.25 ms per token for a 1.3 billion parameter model and 20.9 ms per token for a 66 billion parameter model, outperforming

traditional GPU setups by factors of 2.09 and 1.37, respectively, while demonstrating superior energy efficiency compared to NVIDIA H100 and L4 servers [70, 71, 60, 62].

The automatic INT4 weight-only quantization flow and specialized LLM runtime for optimized CPU performance further illustrate the potential for reducing computational costs and improving inference efficiency [64]. This approach is particularly relevant for deploying LLMs on CPUs, where existing solutions are often optimized for GPU environments.

The simplified LLM decoder layer and segment KV cache policy proposed by Wu et al. improve both latency and throughput during inference by optimizing memory management and reducing redundant memory usage [60]. These innovations are critical for enhancing LLM operations' efficiency, especially in applications requiring rapid response times.

Moreover, the MA-LMM framework demonstrates the effectiveness of memory augmentation by referencing historical video content stored in a memory bank, thereby mitigating limitations of LLMs' context length and reducing GPU memory usage [58]. APEX simulates LLM serving to find optimal parallel execution plans that minimize end-to-end latency, showcasing strategic design choices in optimizing LLM operations [50]. Additionally, CachedAttention employs a hierarchical caching system to enable efficient reuse of KV caches across multi-turn conversations, significantly reducing inference costs and improving prompt prefilling throughput [23].

Furthermore, LLM-Pilot introduces a performance characterization tool that benchmarks LLM inference services under realistic workloads while optimizing the maximum batch weight for each GPU, a novel approach not seen in previous benchmarks [62]. This tool provides valuable insights into optimizing resource allocation and maximizing throughput during LLM inference.

Optimizing inference and training for LLMs involves a comprehensive approach integrating innovative architectural designs, efficient data handling techniques, and advanced optimization methodologies. These strategies are essential for advancing the development and practical application of LLMs across various fields, including information retrieval, scientific literature reviews, ethical considerations, and multimodal representation tasks, thereby enhancing user efficiency and addressing unique challenges in each domain [30, 72, 31, 25].

Feature	Innovative Hardware Strategies	Techniques for Reducing Computational Costs	Optimizing Inference and Training
Optimization Technique	Layernorm Precision	6-bit Quantization	Int4 Quantization
Target Hardware	Fpga	Cpu, Gpu	Cpu
Performance Improvement	Latency Reduction	Memory Usage Reduction	Inference Speed

Table 2: This table provides a comparative analysis of various hardware optimization techniques for large language models (LLMs), highlighting innovative strategies, methods for reducing computational costs, and approaches for optimizing inference and training. It categorizes the techniques based on optimization methods, target hardware, and performance improvements, offering insights into their efficacy and application contexts.

5 LLMs for Hardware Optimization

The intersection of large language models (LLMs) and hardware optimization represents a pivotal research area, demonstrating the transformative impact of AI on system design and performance enhancement. This section explores the diverse applications of LLMs in hardware optimization, starting with AI-driven hardware design processes.

5.1 AI-Driven Hardware Design Processes

Incorporating LLMs into hardware design marks a significant advancement in leveraging AI to improve system efficiency and automate complex tasks. LLMs facilitate real-time dynamic optimization configurations, crucial for executing multilingual tasks effectively [6]. This adaptability underscores LLMs' capability to refine hardware design processes using domain-specific insights.

In healthcare, the MMLLMAIS framework automates the analysis of dental records and imaging data, offering insights and treatment recommendations, showcasing LLMs' versatility in optimizing hardware systems that demand robust interaction capabilities [7]. The Richelieu agent further

exemplifies LLMs' role in enhancing decision-making and negotiation in complex settings, essential for developing adaptive hardware systems responsive to evolving needs [8].

A dataset from real-world production traces of LLM inference requests over 5.5 months provides a realistic depiction of usage patterns, critical for optimizing hardware design processes [62]. This data is vital for ensuring the reliability and efficiency of hardware systems in dynamic environments.

The application of LLMs in AI-driven hardware design signifies a major leap in optimizing hardware systems across various applications. Companies like Google, Amazon, and Apple are investing in advanced hardware accelerators, such as GPUs and FPGAs, to support LLMs' computational demands. This evolution fosters the development of sophisticated AI applications, from natural language processing to complex decision-making systems, laying the groundwork for innovations that could transform multiple sectors. Ongoing enhancements in hardware infrastructure and optimization techniques are crucial for overcoming current limitations and advancing the next generation of AI capabilities [21, 22, 3, 5, 73].

5.2 Optimization Techniques and Frameworks

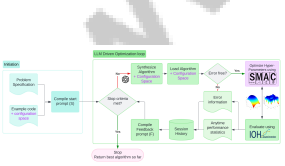
Enhancing hardware performance through LLMs requires a comprehensive approach that integrates advanced frameworks and innovative techniques to boost computational efficiency and adaptability. The LLM-Pilot framework, for example, has achieved a 33

Benchmarking LLMs through a systematic evaluation framework that consolidates a large corpus of Verilog code provides a solid foundation for functional analysis and performance assessment [74]. This approach is crucial for ensuring the reliability and efficiency of hardware systems in applications demanding precise performance metrics.

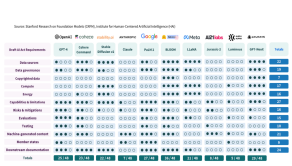
Future research emphasizes bridging LLMs with embodied AI systems, focusing on multimodal integration and addressing the ethical implications of deploying such technologies [73]. This integration is essential for developing hardware systems that are both efficient and ethically sound.

Moreover, expanding frameworks like LLMCompass to support additional machine learning workloads and LLM fine-tuning is a significant step towards enhancing the versatility of LLMs in hardware optimization [75]. This expansion is vital for meeting the evolving demands of AI-driven hardware systems and ensuring their continued relevance.

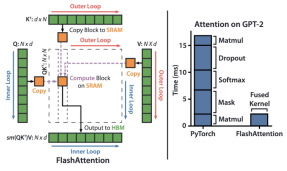
The optimization techniques and frameworks for enhancing hardware performance through LLMs are characterized by innovative resource management approaches, performance benchmarking, and ethical integration. Recent advancements in AI, particularly through LLMs, facilitate the development of efficient and adaptable applications while addressing ethical considerations, such as transparency and accountability. This progress enables diverse applications across various domains, including computing education, where LLMs are reshaping pedagogical approaches and guiding responsible AI development, ultimately enhancing the societal impact of these technologies [5, 31, 18, 73].



(a) LLM-Driven Optimization Loop[76]



(b) Draft AI Act Requirements[77]



(c) FlashAttention: A Fast and Memory-Efficient Attention Mechanism for Large Language Models[78]

Figure 6: Examples of Optimization Techniques and Frameworks

As depicted in Figure 6, exploring LLMs for hardware optimization involves various techniques and frameworks that enhance computational efficiency and performance. The LLM-Driven Optimization Loop illustrates a systematic approach to optimizing hyperparameters using LLMs, beginning with problem specification and example code, leading to algorithm synthesis and evaluation within a defined configuration space, ultimately employing SMAC for performance assessment. The Draft

AI-generated, for reference only.

AI Act Requirements table provides a comprehensive overview of compliance metrics evaluated by Stanford's CRFM and the Institute for Human-Centered Artificial Intelligence, highlighting the regulatory landscape for AI models. Additionally, the FlashAttention mechanism presents a significant advancement, offering a fast and memory-efficient alternative to traditional attention mechanisms in large language models like GPT-2. By comparing FlashAttention with PyTorch's native attention, the figure underscores the potential for reduced computational time and enhanced performance, advocating for the adoption of innovative techniques in hardware optimization for LLMs.

6 AI Agent Construction

6.1 Frameworks and Methodologies for AI Agent Construction

The development of AI agents utilizing large language models (LLMs) hinges on sophisticated frameworks and methodologies that enhance their capacity to comprehend and navigate complex environments. Central to this is the integration of adaptive architectures that enable continuous learning and personalization, allowing AI agents to evolve in line with user preferences and improve interaction quality [24]. In healthcare, leveraging multi-modal data enhances diagnostic precision and reduces clinician workload, underscoring the importance of embedding LLM capabilities into AI agent designs for superior performance across sectors [7]. Moreover, LLM-driven pedagogical innovations have been recognized for enhancing problem-solving skills, thus transforming educational methodologies and fostering conducive learning environments [5].

Future research should prioritize the development of more efficient adaptive architectures for LLMs and explore strategies to mitigate forgetting while integrating external knowledge. These advancements are vital for constructing AI agents that sustain and enhance performance over time, employing methods like document-wise memory architectures and targeted memory retrieval to improve understanding and recall [5, 73, 41]. Frameworks for AI agent construction emphasize adaptability, efficiency, and rigorous validation. Recent AI advancements, particularly through LLMs, facilitate the systematic translation of expert domain knowledge into quantifiable features, enhancing predictive analytics and decision-making across fields. As LLMs progress, they enable the integration of human expertise into machine learning frameworks, unlocking sophisticated applications in diverse areas, including computing education and risk assessment. This synergy between human insight and advanced AI capabilities is driving transformative changes in complex problem-solving and analytics [5, 73, 20].

6.2 Integration of LLMs into AI Systems

Integrating large language models (LLMs) into AI systems is crucial for enhancing functionality and interaction capabilities, leading to more advanced operations. A critical aspect of this integration involves optimizing inference performance, especially in resource-constrained environments like mobile devices. Xiao's research highlights the influence of hardware capabilities and system dynamics on LLM performance, emphasizing the need for tailored integration strategies to maximize efficiency [79].

To enhance efficiency and security, innovative methods, such as Liu's private inference method, replace complex operators with simpler, privacy-friendly functions, accelerating model inference while ensuring data privacy [80]. This approach is essential for integrating LLMs into AI systems where privacy is paramount. Effective integration demands seamless interaction between LLMs and other AI system components, requiring a deep understanding of both LLM architecture and the host system's requirements. By combining LLM capabilities with advanced AI computing systems and human-centered transparency approaches, AI systems can significantly enhance decision-making, optimize user interactions, and execute complex tasks. This integration not only leverages LLMs' natural language understanding to convert expert insights into actionable features for predictive analytics but also addresses overconfidence and knowledge boundaries in LLMs, ensuring more accurate outputs. Furthermore, adopting heterogeneous computing systems is crucial to meet the growing computational demands of LLMs, thereby improving their performance and scalability across applications [41, 21, 18, 81, 20].

The integration of LLMs into AI systems is a complex process requiring careful consideration of hardware capabilities, privacy requirements, and system dynamics. Addressing critical aspects such as hardware optimization, distributed computing strategies, and algorithm design allows AI systems to maximize LLM capabilities, resulting in enhanced functionality and effective interactions across diverse applications, including machine translation and automated literature reviews. This comprehensive approach not only boosts computational efficiency but also bridges performance gaps among various LLMs, facilitating their real-world adoption [30, 25, 21].

7 AI Hardware Integration

7.1 Role of LLMs in AI Hardware Integration

The integration of large language models (LLMs) into AI hardware platforms marks a pivotal advancement in AI, driven by the growing complexity and computational requirements of models like BERT and ChatGPT. These models excel in applications such as machine translation and code generation, necessitating specialized hardware accelerators, including GPUs, TPUs, and MLUs. Consequently, heterogeneous computing systems optimized for distributed environments are essential to manage their increasing memory and computational demands. Recent studies highlight the need for platform designs that accommodate diverse LLM workloads, ensuring efficient deployment of future models potentially exceeding hundreds of trillions of parameters [21, 22]. LLMs enhance hardware platforms by optimizing processing efficiency and interaction capabilities, crucial for real-time processing and decision-making applications.

LLMs significantly contribute to AI hardware integration by optimizing hardware performance through intelligent data processing and task management. The EdgeLLM framework, for example, demonstrates how LLMs can enhance AI task execution on resource-constrained edge devices [3]. This integration not only boosts computational efficiency but also facilitates sophisticated AI functionalities in decentralized settings, expanding AI technologies' applicability.

Moreover, LLMs enable adaptive hardware systems that dynamically respond to varying conditions and user needs, essential for AI systems operating in diverse environments. Their real-time data processing and analysis capabilities translate expert intuition into quantifiable features, improving decision-making accuracy in complex scenarios. LLM-infused applications prioritize transparency, ensuring stakeholders can understand and utilize these advanced systems effectively [18, 25, 20].

Furthermore, integrating LLMs into AI hardware platforms enhances system security and robustness. Leveraging LLMs' advanced language understanding, AI systems can implement sophisticated security measures, such as interactive honeypots, to detect and analyze malicious activities. This innovative approach strengthens cybersecurity infrastructure while integrating expert insights into predictive analytics, enhancing risk assessment and decision-making accuracy [20, 15, 21, 1]. This is particularly critical in fields prioritizing data privacy and security, like healthcare and finance.

Integrating LLMs into AI hardware systems significantly enhances AI applications' functionality, efficiency, and security, as these models require substantial computational resources and optimized hardware platforms for intricate training and inference processes. Utilizing advanced accelerators like GPUs, TPUs, and MLUs, alongside developing heterogeneous computing systems designed for distributed environments, addresses LLMs' growing computational and memory demands. These advancements pave the way for improved performance across applications, from chatbots and code generation to scientific predictions and beyond [21, 22]. Integrating LLMs into hardware platforms enables AI technologies to achieve higher performance and adaptability, fostering more advanced AI solutions across diverse applications.

7.2 Solutions for Effective Integration

Effective integration of large language models (LLMs) into AI hardware systems requires addressing challenges related to computational efficiency, scalability, and resource management. Optimizing hardware architecture is crucial for supporting LLM operations, exemplified by the SECDA-LLM platform, which streamlines FPGA-based LLM accelerators' design and deployment, enhancing performance and reducing latency in inference tasks [19]. Such optimization is vital for embedding AI functionalities into hardware systems demanding high throughput and low latency.

Advanced quantization techniques are another strategy to alleviate LLMs' computational burden. The TC-FPx method, supporting flexible quantization of float-point weights across various bit-widths, illustrates quantization's potential to enhance hardware performance without sacrificing model accuracy [59]. By reducing memory usage and improving computation speed, these techniques enable more efficient LLM integration into hardware platforms, especially those with limited resources.

Near-storage processing, as demonstrated by the Smart-Infinity framework, offers an innovative solution to minimize data transfer and accelerate training processes [54]. Conducting parameter updates closer to storage reduces data movement overhead, enhancing LLM operations' overall efficiency within hardware systems.

Additionally, developing adaptive software frameworks facilitating seamless interaction between LLMs and hardware components is vital for effective integration. The AutoFlow framework, utilizing fine-tuning-based and in-context-based methods for workflow generation, underscores adaptable software solutions' importance in optimizing LLM operations across diverse hardware configurations [17]. Such frameworks ensure LLMs' efficient deployment on various hardware platforms, meeting diverse application requirements.

Moreover, advanced memory management techniques optimize resource usage. The segment KV cache policy proposed by Wu et al. enhances memory efficiency by reducing redundant memory usage and improving throughput during inference [60]. These memory optimization strategies are critical for embedding AI functionalities into hardware systems, particularly in environments with constrained memory resources.

To effectively integrate LLMs into AI hardware systems, a multifaceted approach is essential, encompassing hardware optimization techniques, advanced quantization methods, near-storage processing capabilities, adaptive software frameworks, and robust memory management strategies. This integration is crucial for addressing LLMs' increasing computational and memory demands, necessitating specialized accelerators like GPUs, TPUs, and MLUs, alongside heterogeneous computing systems optimized for distributed environments. By focusing on these areas, developers can significantly enhance LLM deployments' performance and efficiency across various applications [82, 21, 22]. Collectively, these strategies address AI functionalities' embedding challenges into hardware, enabling the development of more efficient, scalable, and versatile AI systems.

8 Computational Efficiency

8.1 Enhancing Computational Efficiency and Resource Optimization

Optimizing computational efficiency and resource utilization in AI systems with large language models (LLMs) is crucial for scalable deployments across diverse domains. The EdgeLLM framework exemplifies dynamic model compilation and execution on edge devices, enhancing data handling and extending AI capabilities in decentralized environments, which is particularly beneficial in resource-constrained edge computing scenarios [3]. The SLaNC method addresses numerical challenges in LayerNorm under low-precision formats, ensuring low latency and high accuracy during inference, thereby optimizing computational efficiency in LLM operations even under stringent precision demands [51].

The Galois system demonstrates the potential of hybrid query execution environments, executing SQL queries on LLMs with accuracy comparable to traditional database management systems, highlighting the importance of optimizing computational processes to balance accuracy and resource utilization [2]. In video processing, the MA-LMM framework reduces GPU memory usage while enabling effective long-term video understanding, emphasizing the significance of memory management in resource optimization [58].

Additionally, the Richelieu agent's self-evolving nature enhances computational efficiency and resource optimization by enabling adaptability and improvement without human intervention [8]. This adaptability underscores the potential for AI systems to dynamically optimize resource allocation, ensuring efficient operation across diverse environments. Leveraging methodologies such as dynamic compilation, precision optimization, hybrid query execution, and adaptive learning frameworks is crucial for harnessing LLM capabilities effectively, as demonstrated in applications like automating literature reviews and enhancing information retrieval [30, 41, 25, 20].

8.2 Algorithmic Optimizations

Algorithmic optimizations are critical for enhancing the efficiency of LLM operations, focusing on strategies that improve computational performance and resource management. The Infinite-LLM framework exemplifies adaptive resource management based on dynamic LLM request needs, ensuring high throughput and efficient memory utilization in real-time applications [83]. Dynamic width speculative beam decoding maintains high-quality outputs while achieving faster inference times by adjusting beam width dynamically, optimizing the decoding process and enhancing overall LLM operational efficiency [84].

Transforming natural language to task language highlights the importance of precise algorithmic metrics in evaluating model performance, focusing on binary accuracy to optimize the transformation process and improve LLM output reliability [28]. These algorithmic optimizations, encompassing adaptive resource management, dynamic decoding strategies, and precise evaluation metrics, are crucial for improving LLM deployment efficiency across various applications. Prioritizing transparency, ethical considerations, and optimized computing infrastructure ensures LLMs effectively meet stakeholder needs while addressing challenges like accountability, bias, and resource management [31, 18, 21].

8.3 Hardware-Software Co-Design

Hardware-software co-design is essential for achieving optimal performance in systems utilizing LLMs. This approach integrates hardware and software components to create efficient and scalable systems that meet LLM computational demands. The SECDA-LLM platform exemplifies this by streamlining the deployment of FPGA-based LLM accelerators, enhancing performance and reducing latency [19]. This illustrates the potential of co-design strategies to optimize hardware resources and improve LLM inference efficiency, achieving higher throughput and lower latency critical for real-time applications.

The Smart-Infinity framework demonstrates the importance of optimizing data flow between hardware and software components through near-storage processing devices, minimizing data transfer overhead and accelerating processing times [54]. Developing specialized runtime environments tailored for optimized CPU performance emphasizes aligning software frameworks with specific hardware capabilities [64]. These environments enable efficient LLM deployment across various hardware configurations, ensuring effective utilization of both hardware and software components.

Hardware-software co-design is crucial for enhancing the performance of LLM-utilizing systems, promoting an integrated approach that harmonizes hardware and software elements. Co-design strategies facilitate the creation of AI systems that are efficient, scalable, and adaptable to the complex demands of contemporary applications. Understanding the interplay between platform design and application requirements is essential for developing high-performing, context-aware AI solutions that enhance productivity in real-world scenarios as AI technologies continue to evolve [22, 63, 73, 35, 21].

9 Conclusion

9.1 Impact on Real-World Applications

The integration of large language models (LLMs) with hardware optimization has significantly transformed various real-world applications, enhancing efficiency and effectiveness across diverse sectors. In sentiment analysis, the implementation of advanced optimization techniques has notably improved model accuracy and reduced error rates, highlighting the potential of LLMs to refine analytical processes. In autonomous driving, the deployment of multimodal LLMs has improved decision-making capabilities, enabling the development of sophisticated systems adept at navigating complex environments. Cybersecurity has also benefited from LLM-driven advancements, with enhanced honeypot systems providing deeper insights into attacker behaviors and fortifying defense mechanisms. In healthcare, applications like ChatGPT are revolutionizing diagnostic and treatment planning processes, offering personalized solutions that cater to individual patient needs. Educational practices are being reshaped through the integration of LLMs, fostering innovative teaching methodologies and interactive learning experiences. Furthermore, the EdgeLLM framework's improvements

in throughput and energy efficiency underscore its relevance in edge computing, where resource management is critical. Collectively, these advancements underscore the transformative impact of LLMs and hardware optimization, necessitating ongoing exploration to address the complexities and ethical considerations inherent in their deployment.

9.2 Future Directions

The future of LLMs and hardware optimization promises substantial advancements through focused research aimed at enhancing functionality and broadening application scopes. Expanding datasets and validating context comprehension at the paragraph level will be crucial, particularly in fields requiring precise understanding, such as legal and regulatory domains. Integrating traditional methods with LLM capabilities in ontology learning could further advance automatic knowledge extraction and representation. Efforts to enhance language diversity by incorporating a wider array of linguistic data into pre-trained models will improve performance in multilingual contexts. Investigating scalability in translation models and optimizing vocabulary size for better outcomes are essential areas for further inquiry. Enhancements to frameworks like the D2LLM, focusing on interaction emulation and parameter-efficient training, are expected to boost performance. In video processing, hierarchical methods for handling long videos and improving visual encoders warrant attention. Optimizing caching strategies and exploring scalable methods like CachedAttention could reduce inference costs and improve efficiency. In computing education, developing instructional strategies that integrate LLMs while addressing ethical concerns is vital for advancing teaching methodologies. Extending AI frameworks to accommodate complex social interactions could offer insights into developing sophisticated AI agents. Future research should also prioritize enhancing LLM robustness, exploring automated optimization techniques, and investigating interdisciplinary applications that leverage the synergy between LLMs and hardware optimization. These directions highlight the potential for developing more sophisticated, efficient, and versatile AI systems across a wide range of applications.

References

[1] Hakan T. Otal and M. Abdullah Canbaz. Llm honeypot: Leveraging large language models as advanced interactive honeypot systems, 2024.

[2] Mohammed Saeed, Nicola De Cao, and Paolo Papotti. Querying large language models with sql, 2023.

[3] Mingqiang Huang, Ao Shen, Kai Li, Haoxiang Peng, Boyu Li, and Hao Yu. Edgellm: A highly efficient cpu-fpga heterogeneous edge accelerator for large language models. *arXiv preprint arXiv:2407.21325*, 2024.

[4] Javier García Gilabert, Carlos Escolano, Aleix Sant Savall, Francesca De Luca Fornaciari, Audrey Mash, Xixian Liao, and Maite Melero. Investigating the translation capabilities of large language models trained on parallel data only, 2024.

[5] James Prather, Paul Denny, Juho Leinonen, Brett A. Becker, Ibrahim Albluwi, Michelle Craig, Hieke Keuning, Natalie Kiesler, Tobias Kohn, Andrew Luxton-Reilly, Stephen MacNeil, Andrew Peterson, Raymond Pettit, Brent N. Reeves, and Jaromir Savelka. The robots are here: Navigating the generative ai revolution in computing education, 2023.

[6] Somnath Kumar, Vaibhav Balloli, Mercy Ranjit, Kabir Ahuja, Tanuja Ganu, Sunayana Sitaram, Kalika Bali, and Akshay Nambi. Bridging the gap: Dynamic learning strategies for improving multilingual performance in llms, 2024.

[7] Hanyao Huang, Ou Zheng, Dongdong Wang, Jiayi Yin, Zijin Wang, Shengxuan Ding, Heng Yin, Chuan Xu, Renjie Yang, Qian Zheng, and Bing Shi. Chatgpt for shaping the future of dentistry: The potential of multi-modal large language model, 2023.

[8] Zhenyu Guan, Xiangyu Kong, Fangwei Zhong, and Yizhou Wang. Richelieu: Self-evolving llm-based agents for ai diplomacy, 2024.

[9] Zihan Liu, Ruinan Zeng, Dongxia Wang, Gengyun Peng, Jingyi Wang, Qiang Liu, Peiyu Liu, and Wenhai Wang. Agents4plc: Automating closed-loop plc code generation and verification in industrial control systems using llm-based agents, 2024.

[10] Hongcheng Ding, Xuanze Zhao, Shamsul Nahar Abdullah, Deshinta Arrova Dewi, Zixiao Jiang, and Xiangyu Shi. Dynamic adaptive optimization for effective sentiment analysis fine-tuning on large language models, 2024.

[11] Yizhang Jin, Jian Li, Yexin Liu, Tianjun Gu, Kai Wu, Zhengkai Jiang, Muyang He, Bo Zhao, Xin Tan, Zhenye Gan, Yabiao Wang, Chengjie Wang, and Lizhuang Ma. Efficient multimodal large language models: A survey, 2024.

[12] Hamed Babaei Giglou, Jennifer D'Souza, and Sören Auer. Llms4ol: Large language models for ontology learning, 2023.

[13] Can Cui, Yunsheng Ma, Xu Cao, Wenqian Ye, Yang Zhou, Kaizhao Liang, Jintai Chen, Juanwu Lu, Zichong Yang, Kuei-Da Liao, et al. A survey on multimodal large language models for autonomous driving. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 958–979, 2024.

[14] Yiheng Liu, Tianle Han, Siyuan Ma, Jiayue Zhang, Yuanyuan Yang, Jiaming Tian, Hao He, Antong Li, Mengshen He, Zhengliang Liu, Zihao Wu, Lin Zhao, Dajiang Zhu, Xiang Li, Ning Qiang, Dingang Shen, Tianming Liu, and Bao Ge. Summary of chatgpt-related research and perspective towards the future of large language models, 2023.

[15] Swapnaja Achintalwar, Adriana Alvarado Garcia, Ateret Anaby-Tavor, Ioana Baldini, Sara E. Berger, Bishwaranjan Bhattacharjee, Djallel Bouneffouf, Subhajit Chaudhury, Pin-Yu Chen, Lamogha Chiazor, Elizabeth M. Daly, Kirushikesh DB, Rogério Abreu de Paula, Pierre Dognin, Eitan Farchi, Soumya Ghosh, Michael Hind, Raya Horesh, George Kour, Ja Young Lee, Nishtha Madaan, Sameep Mehta, Erik Miehling, Keerthiram Murugesan, Manish Nagireddy, Inkit Padhi, David Piorkowski, Ambrish Rawat, Orna Raz, Prasanna Sattigeri, Hendrik Strobelt, Sarathkrishna Swaminathan, Christoph Tillmann, Aashka Trivedi, Kush R. Varshney, Dennis

Wei, Shalisha Witherspooon, and Marcel Zalmanovici. Detectors for safe and reliable llms: Implementations, uses, and limitations, 2024.

[16] Shadeeb Hossain, Aayush Gohil, and Yizhou Wang. Using llm such as chatgpt for designing and implementing a risc processor: Execution, challenges and limitations. *arXiv preprint arXiv:2401.10364*, 2024.

[17] Zelong Li, Shuyuan Xu, Kai Mei, Wenyue Hua, Balaji Rama, Om Raheja, Hao Wang, He Zhu, and Yongfeng Zhang. Autoflow: Automated workflow generation for large language model agents, 2024.

[18] Q. Vera Liao and Jennifer Wortman Vaughan. Ai transparency in the age of llms: A human-centered research roadmap, 2023.

[19] Sen Huang, Kaixiang Yang, Sheng Qi, and Rui Wang. When large language model meets optimization, 2024.

[20] Phoebe Jing, Yijing Gao, Yuanhang Zhang, and Xianlong Zeng. Translating expert intuition into quantifiable features: Encode investigator domain knowledge via llm for enhanced predictive analytics, 2024.

[21] Zhen-Xing Zhang, Yuan-Bo Wen, Han-Qi Lv, Chang Liu, Rui Zhang, Xia-Qing Li, Chao Wang, Zi-Dong Du, Qi Guo, Ling Li, et al. Ai computing systems for llms training: a review. *Journal of Computer Science and Technology*.

[22] Abhimanyu Bambhaniya, Ritik Raj, Geonhwa Jeong, Souvik Kundu, Sudarshan Srinivasan, Midhilesh Elavazhagan, Madhu Kumar, and Tushar Krishna. Demystifying platform requirements for diverse llm inference use cases. *arXiv preprint arXiv:2406.01698*, 2024.

[23] Bin Gao, Zhuomin He, Puru Sharma, Qingxuan Kang, Djordje Jevdjic, Junbo Deng, Xingkun Yang, Zhou Yu, and Pengfei Zuo. Cost-efficient large language model serving for multi-turn conversations with cachedattention, 2024.

[24] Zixian Huang, Wenhao Zhu, Gong Cheng, Lei Li, and Fei Yuan. Mindmerger: Efficient boosting llm reasoning in non-english languages, 2024.

[25] Yilun Zhao, Haowei Zhang, Shengyun Si, Linyong Nan, Xiangru Tang, and Arman Cohan. Investigating table-to-text generation capabilities of llms in real-world information seeking scenarios, 2023.

[26] Yannis Bendi-Ouis, Dan Dutartre, and Xavier Hinaut. Deploying open-source large language models: A performance analysis, 2025.

[27] Zihan Liao, Hang Yu, Jianguo Li, Jun Wang, and Wei Zhang. D2llm: Decomposed and distilled large language models for semantic search, 2024.

[28] Yongchao Chen, Rujul Gandhi, Yang Zhang, and Chuchu Fan. Nl2tl: Transforming natural languages to temporal logics using large language models, 2024.

[29] Gaurav Singh and Kavitesh Kumar Bali. Enhancing decision-making in optimization through llm-assisted inference: A neural networks perspective, 2024.

[30] Dmitry Scherbakov, Nina Hubig, Vinita Jansari, Alexander Bakumenko, and Leslie A. Lenert. The emergence of large language models (llm) as a tool in literature reviews: an llm automated systematic review, 2024.

[31] Junfeng Jiao, Saleh Afroogh, Yiming Xu, and Connor Phillips. Navigating llm ethics: Advancements, challenges, and future directions, 2024.

[32] Dimitrios Michael Manias, Ali Chouman, and Abdallah Shami. Semantic routing for enhanced performance of llm-assisted intent-based 5g core network management and orchestration, 2024.

[33] Shabnam Hassani. Enhancing legal compliance and regulation analysis with large language models, 2024.

[34] Usman Anwar, Abulhair Saparov, Javier Rando, Daniel Paleka, Miles Turpin, Peter Hase, Ekdeep Singh Lubana, Erik Jenner, Stephen Casper, Oliver Sourbut, Benjamin L. Edelman, Zhaowei Zhang, Mario Günther, Anton Korinek, Jose Hernandez-Orallo, Lewis Hammond, Eric Bigelow, Alexander Pan, Lauro Langosco, Tomasz Korbak, Heidi Zhang, Ruiqi Zhong, Seán Ó hÉigeartaigh, Gabriel Recchia, Giulio Corsi, Alan Chan, Markus Anderljung, Lilian Edwards, Aleksandar Petrov, Christian Schroeder de Witt, Sumeet Ramesh Motwan, Yoshua Bengio, Danqi Chen, Philip H. S. Torr, Samuel Albanie, Tegan Maharaj, Jakob Foerster, Florian Tramer, He He, Atoosa Kasirzadeh, Yejin Choi, and David Krueger. Foundational challenges in assuring alignment and safety of large language models, 2024.

[35] Gustavo Pinto, Cleidson de Souza, João Batista Neto, Alberto de Souza, Tarcísio Gotto, and Edward Monteiro. Lessons from building stackspot ai: A contextualized ai coding assistant, 2024.

[36] Seongjun Yang, Gibbeum Lee, Jaewoong Cho, Dimitris Papailiopoulos, and Kangwook Lee. Predictive pipelined decoding: A compute-latency trade-off for exact llm decoding, 2024.

[37] Kai Lv, Shuo Zhang, Tianle Gu, Shuhao Xing, Jiawei Hong, Keyu Chen, Xiaoran Liu, Yuqing Yang, Honglin Guo, Tengxiao Liu, Yu Sun, Qipeng Guo, Hang Yan, and Xipeng Qiu. Collie: Collaborative training of large language models in an efficient way, 2023.

[38] Deepak Narayanan, Mohammad Shoeybi, Jared Casper, Patrick LeGresley, Mostofa Patwary, Vijay Anand Korthikanti, Dmitri Vainbrand, Prethvi Kashinkunti, Julie Bernauer, Bryan Catanzaro, Amar Phanishayee, and Matei Zaharia. Efficient large-scale language model training on gpu clusters using megatron-lm, 2021.

[39] Rana Muhammad Shahroz Khan, Pingzhi Li, Sukwon Yun, Zhenyu Wang, Shahriar Nirjon, Chau-Wai Wong, and Tianlong Chen. Portllm: Personalizing evolving large language models with training-free and portable model patches, 2024.

[40] Tian Liang, Zhiwei He, Wenxiang Jiao, Xing Wang, Yan Wang, Rui Wang, Yujiu Yang, Shuming Shi, and Zhaopeng Tu. Encouraging divergent thinking in large language models through multi-agent debate, 2024.

[41] Bumjin Park and Jaesik Choi. Memorizing documents with guidance in large language models, 2024.

[42] Nicholas Lee, Thanakul Wattanawong, Sehoon Kim, Karttikeya Mangalam, Sheng Shen, Gopala Anumanchipalli, Michael W. Mahoney, Kurt Keutzer, and Amir Gholami. Llm2llm: Boosting llms with novel iterative data enhancement, 2024.

[43] V. K. Cody Bumgardner, Aaron Mullen, Sam Armstrong, Caylin Hickey, and Jeff Talbert. Local large language models for complex structured medical tasks, 2023.

[44] Lincoln Murr, Morgan Grainger, and David Gao. Testing llms on code generation with varying levels of prompt specificity, 2023.

[45] Ruyi Gan, Ziwei Wu, Renliang Sun, Junyu Lu, Xiaojun Wu, Dixiang Zhang, Kunhao Pan, Junqing He, Yuanhe Tian, Ping Yang, Qi Yang, Hao Wang, Jiaxing Zhang, and Yan Song. Ziya2: Data-centric learning is all llms need, 2024.

[46] Hanguang Xiao, Feizhong Zhou, Xingyue Liu, Tianqi Liu, Zhipeng Li, Xin Liu, and Xiaoxuan Huang. A comprehensive survey of large language models and multimodal large language models in medicine, 2024.

[47] Yang Yan, Yihao Wang, Chi Zhang, Wenyuan Hou, Kang Pan, Xingkai Ren, Zelun Wu, Zhixin Zhai, Enyun Yu, Wenwu Ou, and Yang Song. Llm4pr: Improving post-ranking in search engine with large language models, 2024.

[48] Liangming Pan, Michael Saxon, Wenda Xu, Deepak Nathani, Xinyi Wang, and William Yang Wang. Automatically correcting large language models: Surveying the landscape of diverse self-correction strategies, 2023.

[49] Jinxin Liu, Shulin Cao, Jiaxin Shi, Tingjian Zhang, Lunyiu Nie, Linmei Hu, Lei Hou, and Juanzi Li. How proficient are large language models in formal languages? an in-depth insight for knowledge base question answering, 2024.

[50] Yi-Chien Lin, Woosuk Kwon, Ronald Pineda, and Fanny Nina Paravecino. Toward high-performance llm serving: A simulation-based approach for identifying optimal parallelism. *arXiv preprint arXiv:2411.17651*, 2024.

[51] Mahsa Salmani, Nikita Trukhanov, and Ilya Soloveychik. Slanc: Static layernorm calibration, 2024.

[52] Bin Hong, Jinze Wu, Jiayu Liu, Liang Ding, Jing Sha, Kai Zhang, Shijin Wang, and Zhenya Huang. End-to-end graph flattening method for large language models, 2024.

[53] Séamus Lankford and Andy Way. Leveraging llms for mt in crisis scenarios: a blueprint for low-resource languages, 2024.

[54] Hongsun Jang, Jaeyong Song, Jaewon Jung, Jaeyoung Park, Youngsok Kim, and Jinho Lee. Smart-infinity: Fast large language model training using near-storage processing on a real system, 2024.

[55] Mengdi Xu, Peide Huang, Wenhao Yu, Shiqi Liu, Xilun Zhang, Yaru Niu, Tingnan Zhang, Fei Xia, Jie Tan, and Ding Zhao. Creative robot tool use with large language models, 2023.

[56] Minju Seo, Jinheon Baek, and Sung Ju Hwang. Rethinking code refinement: Learning to judge code efficiency, 2024.

[57] Seonjin Na, Geonhwa Jeong, Byung Hoon Ahn, Jeffrey Young, Tushar Krishna, and Hyesoon Kim. Understanding performance implications of llm inference on cpus. In *2024 IEEE International Symposium on Workload Characterization (IISWC)*, pages 169–180. IEEE, 2024.

[58] Bo He, Hengduo Li, Young Kyun Jang, Menglin Jia, Xuefei Cao, Ashish Shah, Abhinav Shrivastava, and Ser-Nam Lim. Ma-lmm: Memory-augmented large multimodal model for long-term video understanding, 2024.

[59] Haojun Xia, Zhen Zheng, Xiaoxia Wu, Shiyang Chen, Zhewei Yao, Stephen Youn, Arash Bakhtiari, Michael Wyatt, Donglin Zhuang, Zhongzhu Zhou, et al. Fp6-llm: Efficiently serving large language models through fp6-centric algorithm-system co-design. *arXiv preprint arXiv:2401.14112*, 2024.

[60] Hui Wu, Yi Gan, Feng Yuan, Jing Ma, Wei Zhu, Yutao Xu, Hong Zhu, Yuhua Zhu, Xiaoli Liu, Jinghui Gu, et al. Efficient llm inference solution on intel gpu. *arXiv preprint arXiv:2401.05391*, 2023.

[61] Alonso Silva. Large language models playing mixed strategy nash equilibrium games, 2024.

[62] Malgorzata Lazuka, Andreea Anghel, and Thomas Parnell. Llm-pilot: Characterize and optimize performance of your llm inference services. In *SC24: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–18. IEEE, 2024.

[63] Jared Fernandez, Luca Wehrstedt, Leonid Shamis, Mostafa Elhoushi, Kalyan Saladi, Yonatan Bisk, Emma Strubell, and Jacob Kahn. Hardware scaling trends and diminishing returns in large-scale distributed training, 2024.

[64] Haihao Shen, Hanwen Chang, Bo Dong, Yu Luo, and Hengyu Meng. Efficient llm inference on cpus. *arXiv preprint arXiv:2311.00502*, 2023.

[65] Yu Zhang, Mingzi Wang, Lancheng Zou, Wulong Liu, Hui-Ling Zhen, Mingxuan Yuan, and Bei Yu. Mixpe: Quantization and hardware co-design for efficient llm inference. *arXiv preprint arXiv:2411.16158*, 2024.

[66] Yeonhong Park, Jake Hyun, SangLyul Cho, Bonggeun Sim, and Jae W. Lee. Any-precision llm: Low-cost deployment of multiple, different-sized llms, 2024.

[67] Yiyuan He, Minxian Xu, Jingfeng Wu, Wanyi Zheng, Kejiang Ye, and Chengzhong Xu. Uellm: A unified and efficient approach for llm inference serving. *arXiv preprint arXiv:2409.14961*, 2024.

[68] Costas Mavromatis, Petros Karypis, and George Karypis. Pack of llms: Model fusion at test-time via perplexity optimization, 2024.

[69] Pei-Fu Guo, Ying-Hsuan Chen, Yun-Da Tsai, and Shou-De Lin. Towards optimizing with large language models, 2024.

[70] Lpu: A latency-optimized and hig.

[71] Keivan Alizadeh, Seyed Iman Mirzadeh, Dmitry Belenko, S Khatamifard, Minsik Cho, Carlo C Del Mundo, Mohammad Rastegari, and Mehrdad Farajtabar. Llm in a flash: Efficient large language model inference with limited memory. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12562–12584, 2024.

[72] Chao Zhang, Haoxin Zhang, Shiwei Wu, Di Wu, Tong Xu, Yan Gao, Yao Hu, and Enhong Chen. Notellm-2: Multimodal large representation models for recommendation. *arXiv preprint arXiv:2405.16789*, 2024.

[73] Frank Joublin, Antonello Ceravola, Joerg Deigmoeller, Michael Gienger, Mathias Franzius, and Julian Eggert. A glimpse in chatgpt capabilities and its impact for ai research, 2023.

[74] Shailja Thakur, Baleegh Ahmad, Zhenxing Fan, Hammond Pearce, Benjamin Tan, Ramesh Karri, Brendan Dolan-Gavitt, and Siddharth Garg. Benchmarking large language models for automated verilog rtl code generation, 2022.

[75] Hengrui Zhang, August Ning, Rohan Prabhakar, and David Wentzlaff. A hardware evaluation framework for large language model inference. *arXiv preprint arXiv:2312.03134*, 2023.

[76] Niki van Stein, Diederick Vermetten, and Thomas Bäck. In-the-loop hyper-parameter optimization for llm-based automated design of heuristics, 2024.

[77] Abi Aryan, Aakash Kumar Nain, Andrew McMahon, Lucas Augusto Meyer, and Harpreet Singh Sahota. The costly dilemma: Generalization, evaluation and cost-optimal deployment of large language models, 2023.

[78] Youngsuk Park, Kailash Budhathoki, Liangfu Chen, Jonas Kübler, Jiaji Huang, Matthäus Kleindessner, Jun Huan, Volkan Cevher, Yida Wang, and George Karypis. Inference optimization of foundation models on ai accelerators, 2024.

[79] Jie Xiao, Qianyi Huang, Xu Chen, and Chen Tian. Large language model performance benchmarking on mobile platforms: A thorough evaluation, 2024.

[80] Xuanqi Liu and Zhuotao Liu. Llms can understand encrypted prompt: Towards privacy-computing friendly transformers, 2023.

[81] Shiyu Ni, Keping Bi, Jiafeng Guo, and Xueqi Cheng. When do llms need retrieval augmentation? mitigating llms' overconfidence helps retrieval augmentation, 2024.

[82] Hao Mark Chen, Wayne Luk, Ka Fai Cedric Yiu, Rui Li, Konstantin Mishchenko, Stylianos I Venieris, and Hongxiang Fan. Hardware-aware parallel prompt decoding for memory-efficient acceleration of llm inference. *arXiv preprint arXiv:2405.18628*, 2024.

[83] Bin Lin, Chen Zhang, Tao Peng, Hanyu Zhao, Wencong Xiao, Minmin Sun, Anmin Liu, Zhipeng Zhang, Lanbo Li, Xiafei Qiu, Shen Li, Zhigang Ji, Tao Xie, Yong Li, and Wei Lin. Infinite-llm: Efficient llm service for long context with distattention and distributed kvcache, 2024.

[84] Zongyue Qin, Zifan He, Neha Prakriya, Jason Cong, and Yizhou Sun. Dynamic-width speculative beam decoding for efficient llm inference, 2024.

Disclaimer:

SurveyX is an AI-powered system designed to automate the generation of surveys. While it aims to produce high-quality, coherent, and comprehensive surveys with accurate citations, the final output is derived from the AI's synthesis of pre-processed materials, which may contain limitations or inaccuracies. As such, the generated content should not be used for academic publication or formal submissions and must be independently reviewed and verified. The developers of SurveyX do not assume responsibility for any errors or consequences arising from the use of the generated surveys.