Journal of Computer Science and Electrical Engineering
Print ISSN: 2663-1938
Online ISSN: 2663-1946
DOI: https://doi.org/10.61784/jcsee3146

THE EVOLUTION OF LARGE MODEL INFERENCE ARCHITECTURES: FROM CENTRALIZED CLOUDS TO DECENTRALIZED ON-DEVICE INTELLIGENCE

ZeYi Luo
Yuan'an No. 1 Senior High School, Yichang 444200, Hubei, China.

Abstract: The proliferation of large-scale AI models, particularly Large Language Models (LLMs), has made inference a critical and resource-intensive workload. This survey provides a comprehensive review of the historical evolution of inference architectures, charting a distinct trajectory from centralized, cloud-native paradigms to fully decentralized, on-device intelligence. We systematically analyze four key architectural epochs: (1) Device-Cloud, (2) Device-Edge-Cloud, (3) Device-Edge, and (4) pure On-Device inference. For each paradigm, we conduct an in-depth examination of its dominant systems, key enabling technologies, and the inherent advantages and limitations that catalyzed the transition to the subsequent stage. Our analysis reveals that this evolution is driven by a persistent set of trade-offs between computational power, latency, data privacy, cost, and energy efficiency. This paper concludes that the future of AI inference lies not in a single monolithic architecture but in a heterogeneous "compute continuum," where workloads are dynamically orchestrated across a spectrum of resources to meet diverse application demands.
Keywords: Large Language Models (LLMs); AI inference; Cloud computing; Edge computing; On-device AI; Distributed systems; Model optimization; Multi-Access Edge Computing (MEC)

1 INTRODUCTION

Large-scale artificial intelligence (AI) models (LLMs) built on complex architectures like the Transformer and trained on petabytes of data, such as BERT GPT-3, and LLaMA [1-4], have demonstrated remarkable capabilities in fields like natural language processing. However, the inference process—using a trained model to make predictions—faces significant computational and memory challenges, especially in resource-constrained and latency-sensitive scenarios [5-6]. While training these models is a monumental one-time effort, inference is an ongoing operational cost that occurs every time a user interacts with an AI-powered service.

The central thesis of this paper is that the history of large model inference architecture is a narrative of progressive decentralization. This evolutionary path begins with the complete centralization of intelligence in the cloud and moves steadily outward, bringing computation closer to the data source to address core challenges such as high latency, data privacy vulnerabilities, prohibitive operational costs, and network dependency [7-8]. We structure this review around four distinct architectural epochs that delineate this journey:

Epoch 1: Device-Cloud Inference: End-user devices offload the entire inference task to powerful, centralized data centers.

Epoch 2: Device-Edge-Cloud Collaborative Architectures: An "edge" computing layer is introduced to mitigate the latency and bandwidth constraints of the pure cloud approach.

Epoch 3: Device-Edge Architectures: A complete decoupling from the public cloud, confining data and computation to a local environment for maximum privacy and responsiveness.

Epoch 4: On-Device Inference: The entire inference process is executed self-sufficiently on the end-user's device.

This analysis culminates in a forward-looking discussion of a unified, adaptive "compute continuum," where inference workloads are dynamically orchestrated across the full spectrum of available resources.

2 THE ERA OF CENTRALIZED INTELLIGENCE: DEVICE-CLOUD INFERENCE

As the computational capabilities of early consumer devices fell far short of the demands of large models, centralizing the inference workload in powerful data centers (the Device-Cloud architecture) became the only viable solution [9]. In this model, the end-user device acts as a "thin client," capturing input data (e.g., text, voice, images) and sending it over a network to a remote server for processing, after which the result is returned.

2.1 Dominant Architectures and Systems

The Device-Cloud model is primarily implemented through two service architectures:

Infrastructure-as-a-Service (IaaS): Cloud providers like AWS, Azure, and GCP offer on-demand access to raw computational resources, typically involving renting virtual machines (VMs) provisioned with high-performance Graphics Processing Units (GPUs) or Tensor Processing Units (TPUs) [10]. This approach grants organizations maximum control but demands significant expertise in infrastructure management.

Model-as-a-Service (MaaS): Representing a higher level of abstraction, MaaS providers like OpenAI host pre-trained, large-scale models and make them accessible via simple APIs [11]. This model democratizes access to state-of-the-art AI by eliminating the need for users to manage the underlying hardware or software infrastructure.

2.2 Key Enabling Technologies

The viability of this paradigm rests on a stack of technologies designed for large-scale computation:
High-Performance Accelerators: At the heart of cloud AI are specialized hardware accelerators, with NVIDIA GPUs being the de facto standard.
Virtualization and Containerization: GPU virtualization allows a single physical GPU to be partitioned into multiple virtual GPUs (vGPUs). Technologies like Docker and orchestration platforms like Kubernetes are essential for deploying and managing models at scale [12].
Throughput Optimization via Batching: A primary economic goal of cloud inference is maximizing throughput. Batching involves grouping multiple independent requests and processing them as a single batch. This significantly improves computational efficiency, although it increases latency for individual requests.

2.3 Advantages and Inherent Limitations

The advantage of the Device-Cloud architecture lies in its unparalleled scalability and computational power, along with simplified management (especially with MaaS) [13]. However, its inherent limitations spurred the evolution of the next generation of architectures:
High and Unpredictable Latency: The physical distance between the user and the data center imposes a significant lower bound on latency, making it unsuitable for real-time applications like autonomous driving or augmented reality [14].
Data Privacy Risks: Transmitting potentially sensitive user data to a third-party cloud creates significant privacy vulnerabilities and regulatory hurdles (e.g., GDPR, HIPAA) [15].
High Operational Costs: The pay-per-use model of cloud inference can lead to substantial and unpredictable operational expenditures as an application scales.
Network Dependency: The entire service model is predicated on a stable, high-bandwidth internet connection, failing completely in environments with poor connectivity [16].

3 THE INTERMEDIATE STEP: DEVICE-EDGE-CLOUD COLLABORATIVE ARCHITECTURES

In response to the limitations of the centralized cloud model, an intermediate compute tier was introduced: the "edge". In the Device-Edge-Cloud collaborative architecture, the edge computing node acts as a local proxy between end-user devices and the distant cloud data center, aiming to balance the low-latency benefits of the edge with the massive computational scale of the cloud [17]. A prime example is Multi-access Edge Computing (MEC), which places compute resources within the mobile network (e.g., at cellular base stations) [18].

3.1 Core Technical Enablers

The core technologies of this architecture include:
Model Partitioning (Split Computing): A single neural network model is split, with one part executing on the device and the other on the edge or cloud [19-20]. The device executes the initial layers, sending a smaller intermediate representation to the edge/cloud. The selection of the "split point" is a critical optimization problem that balances on-device computation with network transmission load [21].
Dynamic Computation Offloading: The system dynamically decides whether to process a task locally or offload it to the edge/cloud based on real-time conditions like network bandwidth, server load, and device battery level. These decision engines often employ techniques like deep reinforcement learning (DRL) to learn optimal offloading policies [22].
Adaptive Inference with Early Exits: By adding lightweight classifiers ("exits") to the intermediate layers of the network, the model can terminate inference early for simple inputs, producing a confident prediction without executing the full network. This significantly reduces average latency by tailoring the computational depth to input complexity.

3.2 Merits and Persistent Challenges

This three-tier architecture successfully reduced latency and improved bandwidth efficiency [23]. However, it also introduced new challenges, primarily the orchestration complexity of managing computation and data flow across three heterogeneous tiers [24]; the resource constraints of edge servers, which are far less powerful than cloud data centers; and the data privacy issues that persist when data is offloaded to a third-party edge server.

4 DECOUPLING FROM THE CLOUD: THE DEVICE-EDGE PARADIGM

To address the persistent challenges of the three-tier model, a more streamlined, two-tier architecture emerged: the Device-Edge paradigm. In this model, the public cloud is removed from the real-time inference loop, and all computation occurs within a localized environment, distributed between the end-user's device and a private edge server or cluster [25].

This architecture is ideal for scenarios where data sovereignty, security, and ultra-low latency are paramount, such as industrial automation or smart healthcare [26]. Its key technologies include advanced model partitioning and pipeline parallelism to maximize system throughput [27]. Furthermore, because data remains local, this architecture naturally supports privacy-preserving techniques like Federated Learning, where models are trained on decentralized data without the data ever leaving the device [28].
Its main advantages are maximum data privacy and security and ultra-low, predictable latency due to communication over a Local Area Network (LAN) [16]. However, its constraints are also clear: the system's computational capacity is limited by the physically deployed hardware, leading to limited and inelastic scalability, and it requires higher Capital Expenditure (CapEx) and management and maintenance overhead.

5 THE ULTIMATE DECENTRALIZATION: PURE ON-DEVICE INFERENCE

The culmination of the decentralization trend is pure On-Device inference, where the entire AI model is executed self-sufficiently on the end-user's device. This requires a suite of aggressive model optimization techniques to compress the model to fit within the limited resources of mobile devices .

4.1 Foundational Optimization Technologies

The implementation of on-device inference relies on the following key technologies:
Model Quantization: Reducing the numerical precision of model parameters from 32-bit floating-point to 8-bit integers. This reduces model size and memory usage by nearly 4x and leverages the efficiency of integer arithmetic on mobile processors.
Network Pruning: Removing redundant parameters or structures (e.g., filters, channels) from a neural network to create a smaller, more efficient model without significant accuracy loss.
Knowledge Distillation: Training a compact "student" model to mimic the behavior of a much larger "teacher" model, thereby transferring the nuanced knowledge from the large model into the smaller network.
Lightweight Model Architectures: Designing computationally efficient networks from the ground up, such as MobileNet and MobileNetV2, which use efficient building blocks like depthwise separable convolutions to replace expensive standard convolutional layers.

4.2 Frameworks and Hardware

This field is dominated by machine learning frameworks designed for mobile, such as Google's TensorFlow Lite (TFLite) and Apple's Core ML. These frameworks work in conjunction with specialized hardware integrated into modern SoCs, such as Neural Processing Units (NPUs), to maximize performance and energy efficiency. Frameworks like PyTorch and MXNet also provide tools to support mobile deployment.

4.3 Advantages and Fundamental Bottlenecks

The advantages of on-device inference are clear: zero network latency, absolute data privacy, and complete offline functionality. However, it also faces fundamental bottlenecks, including the severe resource constraints of mobile devices in terms of compute power, memory, storage, and thermal dissipation, as well as the high energy consumption and reduced battery life caused by AI computations [29].

6 CONCLUSION AND FUTURE DIRECTIONS

6.1 Synthesis of the Evolutionary Trajectory

From Device-Cloud to On-Device, the historical evolution of inference architectures has been a search for balance among competing objectives of computational power, latency, privacy, and cost. No single architecture is a silver bullet. The future lies in a hybrid, adaptive framework that treats the entire spectrum of resources—from the user's device to the centralized cloud—as a unified "compute continuum" [30].
In this paradigm, intelligent orchestration systems will dynamically route each inference request to the most appropriate execution venue based on factors such as query complexity, Service Level Objectives (SLOs), data privacy requirements, real-time system state (network bandwidth, device battery, server load), and economic cost.

6.2 Emerging Trends and Open Research Questions

Purpose-Built Inference Hardware: The market is shifting towards developing hardware specifically engineered for inference, prioritizing latency, energy efficiency, and cost-per-query over raw training throughput. This includes novel chip designs and domain-specific architectures.
Sustainable and Energy-Aware AI: The exponential growth of AI is creating a significant energy demand. Future research will increasingly focus on energy-aware inference, developing scheduling algorithms and hardware that optimize for performance-per-watt.

Inference for Long-Context and Multi-Modal Models: The Key-Value (KV) cache size of the latest large models scales linearly with context length, posing a significant challenge for the limited memory of edge and on-device systems. Developing novel caching and attention mechanisms is a major open research area.
Security and Trust in Decentralized Systems: While decentralization enhances user privacy, it also introduces a more complex attack surface. Securing collaborative inference systems against adversarial attacks (e.g., data poisoning, model inversion) and ensuring model integrity are critical challenges.
In conclusion, the evolution of large model inference architectures is an ongoing process. The journey from centralization to decentralization reflects a maturing understanding of the complex interplay between computation, performance, privacy, and cost. The future is a continuum, where intelligence will be fluidly and adaptively distributed to deliver the benefits of AI efficiently, securely, and sustainably.

COMPETING INTERESTS

The authors have no relevant financial or non-financial interests to disclose.

REFERENCES

[1] Devlin J, Chang M, Lee K, et al. BERT: Pre-training of deep bidirectional transformers for language understanding. North American Chapter of the Association for Computational Linguistics, 2019.
[2] Brown TB, Mann B, Ryder N, et al. Language models are few-shot learners. ArXiv, 2020, abs/2005.14165.
[3] Touvron H, Lavril T, Izacard G, et al. LLaMA: Open and efficient foundation language models. ArXiv, 2023, abs/2302.13971.
[4] Sun M, Han R, Jiang B, et al. A survey on large language model-based agents for statistics and data science. ArXiv, 2024, abs/2412.14222.
[5] Zhou Z, Ning X, Hong K, et al. A survey on efficient inference for large language models. ArXiv, 2024, abs/2404.14294.
[6] Chang Z, Liu S, Xiong X, et al. A survey of recent advances in edge-computing-powered artificial intelligence of things. IEEE Internet of Things Journal, 2021, 8: 13849-13875.
[7] Kachris C. A survey on hardware accelerators for large language models. ArXiv, 2024, abs/2401.09890.
[8] Li E, Zhou Z, Chen X. Edge intelligence: On-demand deep learning model co-inference with device-edge synergy. Proceedings of the 2018 Workshop on Mobile Edge Communications, 2018.
[9] Nguyen DC, Ding M, Pathirana PN, et al. Federated learning for Internet of Things: A comprehensive survey. IEEE Communications Surveys & Tutorials, 2021, 23: 1622-1658.
[10] Zhao Z, Fang L, Cai Z, et al. Edge computing: Platforms, applications and challenges. Journal of Computer Research and Development, 2018, 55: 327.
[11] Abadi M, Agarwal A, Barham P, et al. TensorFlow: Large-scale machine learning on heterogeneous distributed systems. ArXiv, 2016, abs/1603.04467.
[12] Li S, Wang H, Xu W, et al. Collaborative inference and learning between edge SLMs and cloud LLMs: A survey of algorithms, execution, and open challenges. ArXiv, 2025, abs/2507.16731.
[13] Kang Y, Hauswald J, Gao C, et al. Neurosurgeon: Collaborative intelligence between the cloud and mobile edge. Proceedings of the Twenty-Second International Conference on Architectural Support for Programming Languages and Operating Systems, 2017.
[14] Zhen R, Li J, Ji Y, et al. Taming the titans: A survey of efficient LLM inference serving. ArXiv, 2025, abs/2504.19720.
[15] Ye H, Li J, Lu Q. Deep reinforcement learning for dependent task offloading in multi-access edge computing. IEEE Access, 2024, 12: 166281-166297.
[16] Zhao Z, Barijough KM, Gerstlauer A. DeepThings: Distributed adaptive deep learning inference on resource-constrained IoT edge clusters. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2018, 37: 2348-2359.
[17] Li E, Zeng L, Zhou Z, et al. Edge AI: On-demand accelerating deep neural network inference via edge computing. IEEE Transactions on Wireless Communications, 2019, 19: 447-457.
[18] Chiang C, Liu P, Wang D, et al. Optimal branch location for cost-effective inference on Branchynet. 2021 IEEE International Conference on Big Data (Big Data), 2021: 5071-5080.
[19] Han S, Mao H, Dally WJ. Deep compression: Compressing deep neural network with pruning, trained quantization and Huffman coding. ArXiv: Computer Vision and Pattern Recognition, 2015.
[20] Hinton GE, Vinyals O, Dean J. Distilling the knowledge in a neural network. ArXiv, 2015, abs/1503.02531.
[21] Howard AG, Zhu M, Chen B, et al. MobileNets: Efficient convolutional neural networks for mobile vision applications. ArXiv, 2017, abs/1704.04861.
[22] Sandler M, Howard AG, Zhu M, et al. MobileNetV2: Inverted residuals and linear bottlenecks. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018: 4510-4520.
[23] Paszke A, Gross S, Massa F, et al. PyTorch: An imperative style, high-performance deep learning library. ArXiv, 2019, abs/1912.01703.

[24] Wang X, Jia W. Optimizing edge AI: A comprehensive survey on data, model, and system strategies. ArXiv, 2025, abs/2501.03265.
[25] Mao Y, You C, Zhang J, et al. A survey on mobile edge computing: The communication perspective. IEEE Communications Surveys & Tutorials, 2017, 19: 2322-2358.
[26] Zhang Q, Yang LT, Chen Z, et al. A survey on deep learning for big data. Information Fusion, 2018, 42: 146-157.
[27] Zhang C, Patras P, Haddadi H. Deep learning in mobile and wireless networking: A survey. IEEE Communications Surveys & Tutorials, 2018, 21: 2224-2287.
[28] Zhou Z, Chen X, Li E, et al. Edge intelligence: Paving the last mile of artificial intelligence with edge computing. Proceedings of the IEEE, 2019, 107: 1738-1762.
[29] Zhang M, Shen X, Cao J, et al. EdgeShard: Efficient LLM inference via collaborative edge computing. IEEE Internet of Things Journal, 2025, 12: 13119-13131.
[30] Fergus P, Chalmers C, Henderson W, et al. Pressure ulcer categorization and reporting in domiciliary settings using deep learning and mobile devices: A clinical trial to evaluate end-to-end performance. IEEE Access, 2023, 11: 65138-65152.