Journal of Information Systems Engineering and Management
2025, 10(61s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Generative AI-based Agentic Applications: A Bifurcated Lifecycle Analysis of Training and Inference Paradigms

Venkata Varaha Chakravarthy Kanumetta

Independent Researcher, USA

ARTICLE INFO

Received: 28 Sept 2025

Revised: 03 Nov 2025

Accepted: 11 Nov 2025

ABSTRACT

This article explores the bifurcated lifecycle of generative AI applications, examining the distinct computational regimes of training and inference phases. The training phase encompasses data acquisition protocols, parameter optimization techniques, and computational requirements for developing large language models and multimodal systems. The inference phase addresses autoregressive decoding mechanisms, latency optimization strategies, throughput maximization approaches, and performance benchmarking across deployment frameworks. Technical challenges, including memory constraints, network architecture bottlenecks, and hardware-software co-design imperatives, are analyzed, along with cost-performance tradeoffs between training and inference workloads. Industry trends reveal a transition from training-dominant to inference-focused hardware, the emergence of application-specific integrated circuits and specialized chiplets, and vertical integration of hardware development pipelines by major technology providers. These developments reshape computational infrastructure requirements across technology sectors while establishing frameworks for next-generation generative AI deployments.

Keywords: Generative Artificial Intelligence, Computational Infrastructure, Large Language Models, Hardware Acceleration, Inference Optimization

1. Introduction and Background

Recent advancements in artificial intelligence have witnessed a paradigm shift from traditional ranking and recommendation systems toward generative AI technologies. While recommendation systems primarily operate by matching users with existing content through supervised learning methods, generative AI applications—particularly Large Language Models (LLMs)—create entirely new content through a fundamentally different computational approach. This distinction manifests most notably in the bifurcated lifecycle that characterizes generative AI: a resource-intensive training phase followed by an operationally complex inference phase. The emergence of few-shot learning capabilities in modern language models has transformed the landscape of natural language processing, enabling systems to perform tasks with minimal task-specific examples rather than requiring extensive labeled datasets for each downstream application [1]. This computational flexibility represents a significant departure from traditional recommendation architectures that typically require extensive domain-specific training.

The dual-phase paradigm of generative AI presents unique computational challenges compared to traditional machine learning systems. Training operates as a massively parallel operation requiring substantial computational resources to optimize billions of parameters across distributed accelerator clusters. In contrast, inference primarily demands low-latency, high-throughput operations to support real-time user interactions. This asymmetry in computational requirements necessitates specialized infrastructure designs that can accommodate both phases efficiently. Research has demonstrated that model performance follows predictable scaling laws with respect to model size, dataset size, and computational budget, suggesting that continued scaling will yield further improvements in capabilities, albeit with increasing computational demands [2]. These scaling relationships suggest

Journal of Information Systems Engineering and Management
2025, 10(61s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

that it needs to understand the differing resource profiles for deployments that concentrate on training versus inference.

The evolution of large language models has been astounding since the advent of transformer architectures, evolving from millions of parameters to hundreds of billions. The rapid enhancement of capabilities has gone beyond text modalities into multimodal generative capabilities; see novel systems being built with advanced functionality for audio and video in addition to image sources. Recent models are demonstrating truly unprecedented capabilities for understanding contexts, producing coherent and cohesive content, and completing complex reasoning tasks. This evolution reflects not merely incremental improvements but qualitative transformations in model capabilities as parameter counts and training data volumes have increased. The empirical findings regarding few-shot learning suggest that sufficiently large language models acquire emergent abilities to perform complex tasks without explicit optimization for those objectives, fundamentally changing the approach to model development and deployment [1].

The implications of this work extend beyond an interesting academic exercise and represent a substantial reshaping of the computational infrastructure needs across the information technology sector industries.

Given that organizations are more frequently deploying generative AI applications, understanding the differing computational complexity of outputs produced in training and inference mode will be increasingly important for implementing adequate solutions that meet business expectations in terms of resource and hardware.

The increased understanding of the various output-based compute requirements cannot come soon enough as the industry continues to accelerate away from a system focus on training mode hardware development, only to global and regional data centers focused on inference solutions and varying deployment scenarios. The mathematical relationships governing compute-optimal training suggest that as models grow larger, the computational requirements for effective training grow superlinearly, while the efficiency of parameter utilization improves, creating complex tradeoffs in system design [2]. These relationships underscore the importance of specialized hardware architectures that can efficiently handle both the massive parallelism of training and the latency-sensitive demands of inference.

Characteristic	Training Phase	Inference Phase
Computational Focus	Parameter optimization across billions of variables using distributed systems	Sequential token generation with emphasis on latency and throughput
Resource Requirements	High-performance accelerator clusters with massive memory bandwidth	Lower latency systems optimized for memory-bound operations
Optimization Goals	Model quality and convergence stability across distributed systems	User responsiveness and cost-effective scaling for concurrent requests

Table 1: Comparison of Training vs. Inference Phases in Generative AI. [1, 2]

2. Training Phase Architecture and Methodologies

Developing generative AI systems demands intricate resource management across data handling, parameter tuning, and computational resource allocation. When building large language models, practitioners must gather varied textual sources covering numerous knowledge domains, languages,

Journal of Information Systems Engineering and Management
2025, 10(61s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

and formats to establish comprehensive linguistic foundations. This necessitates elaborate preprocessing workflows that normalize text encodings, eliminate unwanted artifacts, process multilingual materials, and screen out inappropriate content. Current innovations in data preparation focus heavily on enhancing dataset integrity through duplicate removal, statistical filtering based on perplexity measurements, and contextual metadata enhancement to counter poor-quality training examples. The tokenization approach selected significantly affects both performance outcomes and processing efficiency, with contemporary systems favoring subword tokenization strategies that effectively balance vocabulary scope with representation capabilities. Evidence suggests that unifying diverse natural language processing tasks into standardized formats dramatically enhances cross-task knowledge transfer, indicating that initial preprocessing decisions affect not only training efficiency but also downstream task adaptation [3].

Optimizing parameters within vast neural networks presents fundamental challenges when training generative systems containing billions to trillions of adjustable values. Such extensive models demand tailored optimization approaches addressing gradient instability issues and convergence reliability. Practitioners have found particular success with adaptive optimization techniques that individually adjust learning rates according to parameter-specific gradient patterns over time. Practical implementations now routinely incorporate gradient accumulation methods, mixed numerical precision calculations, and distributed optimizer state management to address memory limitations in large-scale implementations. Evidence demonstrates that thoughtfully designed initialization protocols and carefully structured learning rate progressions substantially impact training stability and ultimate model quality, particularly as architectures grow increasingly expansive. Investigations into resource-optimal training configurations reveal that efficient computational distribution depends on nuanced interactions between model architecture, data characteristics, and training durations—factors that increasingly influence optimization strategy selection as models continue scaling upward [4].

Language understanding methodologies have evolved considerably beyond basic predictive token modeling toward multifaceted training objectives. Contemporary techniques frequently combine masked language modeling, contextual sequence prediction, and contrastive learning approaches to develop nuanced contextual representations. The initial training generally uses self-supervised approaches on large collections of text that establish general linguistic skills without the requirement for labeling data relating to the task at hand. Subsequently, the calibration stage integrates instruction-based fine-tuning and human feedback-based reinforcement learning to promote outputs in line with human expectations. This phased development process builds both fundamental language understanding and practically applicable capabilities aligned with user needs. Research demonstrates that structuring varied language tasks into consistent formats substantially improves cross-domain knowledge application, suggesting architectural consistency benefits alignment strategies across applications [3].

Technique	Implementation Approach	Primary Benefit
Gradient Checkpointing	Selective recomputation of activations during the backward pass	Reduces peak memory usage at the cost of additional computation
ZeRO Optimizer	Partitioning optimizer states and parameters across devices	Eliminates memory redundancy while maintaining computational efficiency
Quantization-Aware Training	Lower-precision representation of weights and activations	Decreases memory requirements with minimal impact on model quality

Table 2: Memory Optimization Techniques for Large-Scale Generative Models.[3, 4]

Journal of Information Systems Engineering and Management
2025, 10(61s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Hardware requirements for training sophisticated generative models continue growing exponentially, driving innovations in specialized computing configurations and distributed processing strategies. Modern training infrastructures employ interconnected accelerator clusters linked through high-capacity networks, enabling coordinated parallel operations. Memory management presents significant challenges, requiring careful distribution of weights, optimization states, gradients, and activation values across multiple computing devices. Engineers increasingly rely on model parallelism, sequence pipeline parallelism, and tensor distribution techniques to overcome individual device limitations. Physical constraints, including power consumption and thermal management, further complicate training environments, with contemporary model development consuming substantial electricity continuously over weeks or months. Performance evaluation increasingly emphasizes efficiency metrics, including throughput measurements, power utilization ratios, and distribution scaling effectiveness. Theoretical frameworks now exist for determining optimal resource allocation based on performance targets, suggesting efficiency-oriented training approaches should adapt dynamically as models scale across different operational ranges [4].

3. Inference Phase Engineering Considerations

Generative AI systems encounter distinct engineering challenges during operational deployment that relate to achieving practical efficiency and responsive interactions with users. Text generation in language models is fundamentally autoregressive decoding, which is the sequential construction of the next text element given the previously generated text elements. This sequential dependency creates processing constraints markedly different from the highly parallelizable nature of model training. Current text generation approaches employ numerous efficiency-enhancing methods, including sophisticated beam search variants, probability-based nucleus sampling, and adjustable temperature parameters to harmonize output creativity with processing costs. Pioneering work has recently advanced speculative generation techniques that anticipate multiple forthcoming elements simultaneously, dramatically speeding content creation by minimizing required model calculations. Such techniques employ smaller helper models proposing candidate continuations, which the primary system then validates collectively, cutting response delays while preserving output standards comparable to traditional sequential generation. This innovation demonstrates how autoregressive limitations can be partially overcome through algorithmic inventiveness, exploiting natural language patterns, creating viable acceleration paths without needing hardware replacements or aggressive model simplification [5]. Implementing these next-generation additional approaches requires a careful balance of competing objectives—creativity variability, logical consistency, and throughput—specific to application context and performance needs.

Strategy	Mechanism	Performance Impact
Speculative Decoding	Smaller draft model proposes tokens for parallel verification	Reduces end-to-end generation latency while maintaining output quality
Paged Attention	Memory-efficient management of key-value caches	Enables handling of longer contexts with limited GPU memory
Continuous Batching	Elimination of synchronization barriers between requests	Increases throughput for variable-length sequences in multi-user environments

Table 3: Inference Acceleration Strategies for Generative AI. [5, 6]

Reducing response time is also an important goal in deploying generative AI platforms, particularly for conversational applications where the perceived quality of the results is contingent on responsive model behavior. In numerical precision reduction, mathematical tasks are converted from high-

Journal of Information Systems Engineering and Management
2025, 10(61s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

precision operations to reduced-precision operations, decreasing memory and processing requirements dramatically and producing results that are usually acceptable in terms of integrity. Teacher-student knowledge transfer compresses expansive source models into streamlined versions, maintaining essential capabilities despite reduced complexity. Systematic network trimming selectively eliminates less significant connections, producing sparser structures that operate more efficiently on specialized computing hardware. Beyond model-level adjustments, system architecture improvements, including request grouping, result storage, and priority-based processing, substantially decrease response times in multi-user scenarios. Investigations have revealed that attention mechanisms—core components in transformer architectures—offer significant optimization potential through advanced memory management approaches that efficiently handle extensive context storage during operation. By reconceptualizing attention processing as fundamentally a memory management challenge, these techniques enable substantial performance gains through more effective hardware utilization, particularly when processing lengthy contexts that would typically exceed available computing resources. These innovations demonstrate how structural insights into model architecture enable specialized system-level enhancements that significantly outperform general-purpose processing frameworks [6]. The cumulative effect of these response time improvements ultimately determines whether generative systems meet user expectations across diverse operational environments.

Maximizing processing capacity for multi-user deployments focuses on handling the largest possible concurrent request volume within fixed computational constraints. Effective request consolidation groups multiple generation tasks to exploit parallel processing capabilities in modern accelerators, substantially increasing overall system capacity while potentially extending individual response times. Adaptive grouping techniques dynamically adjust consolidation patterns based on current processing loads and incoming request characteristics, optimizing resource allocation under fluctuating conditions. Strategic caching stores frequently requested inputs, outputs, and intermediate processing states, reducing computational demands for common interactions. Advanced distribution architectures implement sophisticated workload management and balanced allocation techniques across multiple processing endpoints. Research confirms that continuous request grouping approaches, which eliminate synchronization delays between batched operations, deliver substantial capacity improvements compared to traditional fixed-batch methods. Purpose-built serving platforms for transformer-based systems demonstrate that architecture-aware scheduling and memory optimization techniques significantly enhance processing capacity while maintaining acceptable response times. These platforms utilize element-level scheduling mechanisms dynamically allocating resources across multiple simultaneous requests, enabling efficient hardware utilization even with highly variable sequence lengths and processing patterns [7]. These capacity enhancement approaches determine fundamental scaling characteristics for generative AI deployments, directly influencing operational economics and service limitations.

Practical performance evaluation across implementation frameworks provides essential insights guiding deployment decisions. Comprehensive assessment methodologies measure critical indicators, including response time distribution, request handling capacity under various loads, memory consumption patterns, and energy utilization metrics. These evaluations must address diverse computing environments from data center acceleration platforms to resource-constrained edge devices. Performance differences across model scales, input lengths, and processing configurations reveal important scaling characteristics informing resource planning and capacity management. Comparative studies demonstrate significant efficiency differences between various optimization frameworks, with specialized systems substantially outperforming general-purpose alternatives on identical hardware. Memory-efficient attention mechanisms have proven particularly valuable for operational performance by addressing fundamental bottlenecks in contextual information management within transformer architectures. By implementing non-sequential memory access patterns and granular resource management policies, these approaches enable more effective

Journal of Information Systems Engineering and Management
2025, 10(61s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

hardware utilization and improved scaling characteristics for large language model operation [6]. Distributed serving systems further demonstrate how architecture decisions fundamentally impact operational performance, with specialized scheduling algorithms and resource management techniques substantially outperforming conventional approaches. These systems utilize sophisticated workload analysis and adaptive resource allocation strategies, maximizing hardware utilization while meeting application-specific performance requirements across varying operational conditions [7]. Such holistic performance analysis methodologies guide the selection of appropriate deployment architectures for specific application requirements, ensuring generative AI systems deliver consistent, responsive experiences while maintaining practical economic viability.

4. Technical Challenges and Infrastructure Limitations

Rapidly expanding parameter counts in neural models create severe memory bottlenecks, restricting further growth of generative systems. When architectures surpass billions of adjustable parameters, storage needs for model components, optimization states, gradient information, and activation data overwhelm individual processing units, forcing the adoption of multi-device memory solutions. Several inventive techniques address these constraints: gradient checkpointing sacrifices some computational speed to reduce memory usage by recalculating certain values rather than storing them throughout backward propagation; strategic parameter sharing across embedding layers and transformer segments maintains functional capacity while reducing memory demands; precision-aware training methods enable reduced numerical representation during development and operation phases. Sophisticated memory controllers dynamically shift computational elements between fast local memory and larger system storage based on processing schedules. Notable advancements in distributed optimization demonstrate how partitioning strategies can dramatically enhance memory utilization by distributing optimizer data, gradient information, and model parameters across computing units without wasteful duplication, essentially removing redundant storage while preserving processing efficiency. Through carefully structured implementation phases, these methods support the development of models that would otherwise exceed hardware capabilities, while simultaneously enhancing overall throughput via reduced communication requirements and better parallelization. This ability to maximize memory efficiency without computational penalties marks a crucial development in addressing fundamental tensions between model complexity and hardware limitations, enabling continued advancement despite the physical constraints of individual processors [8]. These memory management approaches directly influence practical scaling boundaries, determining maximum viable model complexity for specific hardware configurations.

Network design emerges as a pivotal constraint in distributed computing environments supporting large-scale generative systems. Coordinating massive models across processor clusters demands high-capacity, minimal-delay connections facilitating efficient information exchange between computational nodes. Conventional network designs struggle with communication patterns typical in distributed neural systems, where simultaneous multi-node exchanges frequently occur during parameter updates. Communication bandwidth needs increase alongside model complexity, training batch size, and cluster expansion, creating significant implementation hurdles for organizations deploying advanced language models. Network competition and congestion measurably impact development speed and learning stability, especially in shared environments where multiple workloads compete for limited network resources. Forward-looking research explores bandwidth-efficient distributed training strategies, including gradient data compression, localized update mechanisms, and topology-conscious parameter distribution to reduce communication volume. Investigations into the environmental impacts of contemporary AI systems identify network infrastructure as a major contributor to both upfront investment costs and ongoing energy expenditure. Studies highlight how purpose-designed network architectures can substantially reduce financial and environmental impacts through streamlined data movement, minimized redundancy,

Journal of Information Systems Engineering and Management
2025, 10(61s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

and task-specific optimizations. These specialized designs address unique communication patterns in distributed AI operations, including collective processing operations, parameter synchronization, and data partitioning approaches that differ markedly from traditional computing traffic patterns [9]. Network architecture considerations increasingly shape purpose-built AI infrastructure designs, influencing facility planning, processing cluster arrangement, and hardware selection decisions beyond performance metrics to include sustainability and long-term economic viability.

Co-Design Dimension	Hardware Innovation	Software Innovation
Computation Architecture	Specialized matrix units optimized for transformer operations	Operator fusion and specialized kernel implementations
Memory Hierarchy	Custom memory subsystems with optimized bandwidth for attention operations	Dynamic tensor offloading between device and system memory
Parallelization Strategy	Purpose-built interconnects for AI-specific communication patterns	Automated parallelization frameworks for distributed execution

Table 4: Hardware-Software Co-Design Approaches for Generative AI. [9, 10]

Integrated hardware-software development has become essential for processing efficiency in generative AI applications. The specific computational characteristics of transformer architectures—featuring memory-intensive operations, irregular computational patterns, and complex data interdependencies—demand specialized processing hardware optimized for these particular workloads. Purpose-built acceleration hardware incorporates dedicated matrix computation units, tailored memory structures, and specialized components handling attention mechanisms. Software frameworks must develop alongside hardware innovations, implementing model-specific enhancements, including operation combining, memory layout optimization, and specialized processing routines leveraging hardware-specific capabilities. Runtime compilation methods dynamically optimize computational sequences based on input characteristics and available resources. This integrated approach extends to system-level considerations, including power distribution, heat management, and connection architectures supporting the unique requirements of AI accelerators. Studies examining automated parallelization systems for distributed neural networks demonstrate the crucial importance of hardware-software integration for processing efficiency. These systems automatically coordinate complex parallel processing strategies across multiple operational dimensions, adapting compilation and execution approaches to specific hardware characteristics and communication structures. By developing systematic parallelization methods addressing both algorithmic structure and hardware capabilities, these approaches achieve substantially improved performance compared to manually optimized implementations, highlighting potential benefits of automated design approaches addressing the increasing complexity of generative AI workloads [10]. The synchronized evolution of hardware and software represents a critical development area addressing computational challenges of large-scale generative AI, particularly as applications transition from research to widespread practical deployment.

Economic tradeoffs between development and operational phases present complex considerations for organizations implementing generative AI applications. Development represents an intensive, periodically recurring capital expense, while operational deployment creates ongoing costs scaling with usage patterns. Resource allocation between these phases significantly impacts both

development timelines and operational economics. Development investments produce model assets with extended operational lifespans, suggesting different optimization priorities compared to operational infrastructure continuously serving production workloads. Optimal resource allocation depends on application-specific factors, including expected usage volume, performance requirements, and model update frequency. More organizations are adopting hybrid infrastructure design, appropriately utilizing specialized high-performance hardware for development and using more economical operation-optimized platforms for service delivery with more effective resource management implications. Research into sustainable AI adoption has revealed some significant possibilities for improving resource use in systematic optimization throughout the lifetime. This research emphasizes that development and operational phases present distinct sustainability challenges, with development typically requiring concentrated processing intensity during shorter periods while operational deployment demands long-term efficiency across distributed environments. The analysis highlights how environmental assessments must consider not only direct energy usage but also embodied energy in manufacturing, infrastructure requirements, and the complete system lifecycle [9]. Automated parallelization approaches further illuminate economic dimensions of infrastructure decisions by demonstrating how intelligent resource allocation significantly improves hardware utilization efficiency. By flexibly adjusting execution strategies dynamically to available resources when appropriate and workload characteristics, these strategies optimize processing to improve efficiency in execution and minimize unused capacity that directly impacts operational costs and environmental implications [10]. The economic performance impacts affect the practicality of the generative AI opportunities at scale, as well as the ability to develop or make improvements in hardware efficiency practices, as well as the operational management practices from economic to environmental performance.

5. Conclusion and Future Directions

The generative AI industry is shifting from training-focused to inference-optimized infrastructure as deployment priorities evolve. Initially, systems emphasized raw computational power for training massive models. Now, as these models move into production, attention has turned to optimizing inference for latency, cost-efficiency, and energy consumption. This transition reflects the maturing lifecycle of generative AI, where training represents periodic high-intensity computation while inference becomes a continuous operational requirement.

Application-specific integrated circuits (ASICs) and specialized chiplets are emerging as key hardware solutions for generative AI workloads. These custom silicon designs incorporate features specifically tailored for transformer-based computations, including dedicated matrix multiplication units, optimized attention mechanisms, and memory systems designed for language model access patterns. The modular chiplet approach allows for combining general-purpose components with specialized accelerators, creating flexible systems optimized for specific workload characteristics while maintaining manufacturing efficiency.

Major technology companies and cloud providers are increasingly pursuing vertical integration of hardware development pipelines. By controlling the entire stack from chip design to system architecture, these organizations implement cross-layer optimizations that deliver performance advantages impossible to achieve through component-level integration alone. This trend creates competitive differentiation in cloud offerings while potentially widening the gap between hyperscale providers and smaller organizations relying on commercial solutions.

Future research directions for generative AI infrastructure will likely focus on novel memory architectures addressing current bandwidth limitations, heterogeneous computing systems combining diverse accelerator types, and programming models that abstract growing hardware complexity. Energy efficiency will become increasingly central as deployment scales grow, driving innovations

across all system levels. Perhaps most importantly, research will explore how model architectures and hardware systems must co-evolve, recognizing that continued advancement requires parallel innovation in both algorithms and infrastructure.

References

[1] Tom B. Brown et al., "Language Models are Few-Shot Learners," ResearchGate, 2020. [Online]. Available: https://www.researchgate.net/publication/341724146_Language_Models_are_Few-Shot_Learners

[2] Jared Kaplan et al., "Scaling Laws for Neural Language Models," arXiv preprint arXiv:2001.08361, 2020. [Online]. Available: https://arxiv.org/abs/2001.08361

[3] Colin Raffel et al., "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer," arXiv:1910.10683 [cs.LG], 2019. [Online]. Available: https://arxiv.org/abs/1910.10683#

[4] Jordan Hoffmann et al., "Training Compute-Optimal Large Language Models," arXiv preprint arXiv:2203.15556, 2022. [Online]. Available: https://arxiv.org/abs/2203.15556#

[5] Charlie Chen et al., "Accelerating Large Language Model Decoding with Speculative Sampling," arXiv preprint arXiv:2302.01318, 2023. [Online]. Available: https://arxiv.org/abs/2302.01318

[6] Woosuk Kwon et al., "Efficient Memory Management for Large Language Model Serving with PagedAttention," arXiv preprint arXiv:2309.06180, 2023. [Online]. Available: https://arxiv.org/abs/2309.06180

[7] Gyeong-In Yu et al., "Orca: A Distributed Serving System for Transformer-Based Generative Models," OSDI '22 Open Access Sponsored by NetApp. [Online]. Available: https://www.usenix.org/conference/osdi22/presentation/yu

[8] Samyam Rajbhandari et al., "ZeRO: Memory Optimizations Toward Training Trillion Parameter Models," arXiv preprint arXiv:1910.02054, 2020. [Online]. Available: https://arxiv.org/abs/1910.02054

[9] Abdulaziz Tabbakh et al., "Towards sustainable AI: a comprehensive framework for Green AI," Springer Nature Link, 2024. [Online]. Available: https://link.springer.com/article/10.1007/s43621-024-00641-4

[10] Lianmin Zheng et al., "Alpa: Automating Inter- and Intra-Operator Parallelism for Distributed Deep Learning," arXiv preprint arXiv:2201.12023, 2022. [Online]. Available: https://arxiv.org/abs/2201.12023