applied sciences

MDPI

Review

A Review on AI Miniaturization: Trends and Challenges

Bin Tang, Shengzhi Du * and Antonie Johan Smith

Department of Electrical Engineering, Tshwane University of Technology, Pretoria 0001, South Africa; 224494157@tut4life.ac.za (B.T.); smithaj@tut.ac.za (A.J.S.)
* Correspondence: dushengzhi@gmail.com

Abstract

Artificial intelligence (AI) often suffers from high energy consumption and complex deployment in resource-constrained environments, leading to a structural mismatch between capability and deployability. This review takes two representative scenarios—energy-first and performance-first—as the main thread, systematically comparing cloud, edge, and fog/cloudlet/mobile edge computing (MEC)/micro data center (MDC) architectures. Based on a standardized literature search and screening process, three categories of miniaturization strategies are distilled: redundancy compression (e.g., pruning, quantization, and distillation), knowledge transfer (e.g., distillation and parameter-efficient fine-tuning), and hardware–software co-design (e.g., neural architecture search (NAS), compiler-level, and operator-level optimization). The purposes of this review are threefold: (1) to unify the "architecture–strategy–implementation pathway" from a system-level perspective; (2) to establish technology–budget mapping with verifiable quantitative indicators; and (3) to summarize representative pathways for energy- and performance-prioritized scenarios, while highlighting current deficiencies in data disclosure and device-side validation. The findings indicate that, compared with single techniques, cross-layer combined optimization better balances accuracy, latency, and power consumption. Therefore, AI miniaturization should be regarded as a proactive method of structural reconfiguration for large-scale deployment. Future efforts should advance cross-scenario empirical validation and standardized benchmarking, while reinforcing hardware–software co-design. Compared with existing reviews that mostly focus on a single dimension, this review proposes a cross-level framework and design checklist, systematizing scattered optimization methods into reusable engineering pathways.

Keywords: AI miniaturization; cloud computing; edge computing; energy efficiency; embedded systems

Academic Editor: Luis Javier Garcia Villalba

Received: 28 August 2025
Revised: 7 October 2025
Accepted: 10 October 2025
Published: 12 October 2025

Citation: Tang, B.; Du, S.; Smith, A.J. A Review on AI Miniaturization: Trends and Challenges. *Appl. Sci.* **2025**, *15*, 10958. https://doi.org/10.3390/app152010958

1. Introduction

1.1. Background

Since its inception, artificial intelligence (AI) has been regarded as a transformative force for human society [1]. From early theoretical exploration in the 20th century to real-world deployment in the 21st century, AI has consistently embodied the pursuit of "machine intelligence" [2]. The public release of OpenAI's large language model, ChatGPT (developed by OpenAI, based on the Generative Pre-trained Transformer, GPT, architecture launched in November 2022) [3], marked the entry of AI into a new era of interactivity and sparked a global wave of technological enthusiasm [1].

AI has already achieved remarkable breakthroughs across multiple domains, including visual recognition [4–6], natural language processing (NLP) [7,8], biomedical science [9],

and scientific computing [10]. However, these advances continue to be accompanied by high resource requirements [10], high deployment costs [11], and increasing system complexity [12], making AI more of an advanced tool for a privileged few, rather than a foundational capability universally accessible to all [13]. Consequently, future priorities should shift from merely improving performance toward lowering the barriers to diffusion and enabling structural accessibility.

The successful deployment of AI depends not only on algorithmic capabilities but also on the underlying infrastructure and computational frameworks [14]. In particular, within the Internet of Things (IoT), more than ten billion devices have already been deployed across smart cities [15,16], industrial automation [17], and healthcare monitoring [18], requiring real-time sensing, analysis, and response [19]. Together, IoT and AI constitute the two foundational pillars of intelligent systems [20].

Nevertheless, traditional centralized computing architectures have increasingly exposed their limitations in terms of latency, bandwidth efficiency, and privacy [18]. Hybrid cloud–edge computing has been proposed as a promising solution and has demonstrated advantages in scenarios such as autonomous driving, smart cities, and intelligent manufacturing [21]. Yet, current AI systems still lack structural adaptability to edge environments [19]. On resource-constrained and power-sensitive devices, AI models often struggle to achieve efficient, reliable, and sustainable on-device inference. This contradiction highlights a core issue: today's AI models exhibit widespread structural redundancy and vulnerability when deployed in resource-limited scenarios.

1.2. Comparison with Existing Reviews

In recent years, a growing body of review studies have focused on the deployment challenges and optimization strategies of AI in cloud and edge computing environments. For example, Shi et al. [18] systematically reviewed the architecture and application scenarios of edge intelligence; Gill et al. [22] concentrated on the integration of federated learning and edge computing within 6G networks; Wang et al. [23] provided a comprehensive survey covering device-side AI model optimization, hardware acceleration, and privacy protection; Dantas et al. [24] summarized the applications of pruning, quantization, and distillation techniques in machine learning; and Liu et al. [25] reviewed the evolution of model compression technologies from early approaches to the large-model era, while discussing future trends.

These works cover multiple levels—from system architecture and communication networks to device-side optimization and model compression—and provide valuable references for understanding AI deployment. However, most of them remain confined to a single technical dimension, such as a specific compression technique or hardware optimization strategy, and lack a holistic perspective that views AI miniaturization as a systemic structural evolution. Moreover, existing reviews often evaluate AI systems primarily from a performance perspective, with limited discussion of the structural contradictions between performance, miniaturization, and deployability. Few distinguish between energy-first and performance-first design trade-offs, and few offer a cross-level systemic viewpoint.

In contrast, this review not only summarizes existing methods but also proposes a methodological framework and design checklist, thereby transforming fragmented optimization approaches into reusable engineering pathways to address the limitations of prior reviews.

This review addresses the critical mismatch between AI capability and deployability, particularly the unsustainable growth of model size, computational demand, and energy cost. The central question is how AI miniaturization strategies can systematically trans-

form this contradiction into actionable engineering pathways under both energy-first and performance-first deployment scenarios.

To contextualize these contributions, Table 1 summarizes representative review studies on AI miniaturization and related topics, highlighting their coverage, main contributions, and limitations.

Table 1. Comparison of representative reviews on AI miniaturization and related topics.

Reference	Source	Coverage Theme	Main Contribution	Limitation
Zhou et al., 2019 [19]	Proc. IEEE	Edge intelligence architectures and applications	Systematic overview of edge intelligence and deployment architectures	Limited focus on model miniaturization strategies
Duan et al., 2023 [22]	IEEE COMST	Federated learning + edge computing	Analysis of collaborative intelligence in 6G and edge networks	Communication-oriented, little emphasis on AI miniaturization
Wang et al., 2025 [23]	ACM CSUR	On-device AI models	Comprehensive review of device-side model optimization and hardware acceleration	Lacks cross-layer/ system-level perspective
Dantas et al., 2024 [24]	Applied Intelligence	Model compression review	Summarized pruning, quantization, distillation, etc.	Algorithm-level focused, limited discussion on system evolution
Liu et al., 2025 [25]	Frontiers in Robotics & AI	Evolution of model compression	Reviewed compression methods from early CNNs to large models	Insufficient analysis of adaptability and energy efficiency
Deng et al., 2020 [26]	IEEE IoT J.	Edge intelligence + AI confluence	Proposed taxonomy of edge intelligence applications, challenges, and future directions	More conceptual, limited discussion on energy efficiency and deployment trade-offs

The main contributions of this review can be summarized as follows:

(i) Systematic comparison of representative computing architectures. We summarize and compare cloud computing, edge computing, fog computing, cloudlet, and MDCs, analyzing their respective advantages and limitations in terms of latency, energy efficiency, cost, and structural adaptability. This provides insights into deployment constraints under different computing paradigms;

(ii) Distillation of three core miniaturization strategies. We categorize existing scattered research into three strategic approaches—redundancy compression, knowledge transfer, and hardware–software co-design—and organize representative methods into a unified classification framework;

(iii) Proposal of three structural design principles. Inspired by the development of embedded systems, we propose three principles for AI miniaturization: reducing the execution burden on end devices, enhancing native computational capability through hardware–software co-design, and balancing local intelligence with centralized AI. This emphasizes that miniaturization is not merely model compression but structural reconfiguration;

(iv) Construction of a practice-oriented design framework. Building on the architectural comparison, strategic pathways, and structural principles, we propose a practice-oriented framework for AI miniaturization, offering methodological references for deployment under both energy-first and performance-first scenarios.

1.3. Methodology

To ensure the systematicity and reproducibility of this review, the study follows the general protocols of systematic reviews. Regarding database selection, the relevant literature was retrieved from Scopus, IEEE Xplore, ACM Digital Library, and Web of Science within the time frame of 2015 to June 2025.

Scope of the review. In total, 120 publications (2015–2025) were systematically reviewed, including 35 on model compression, 28 on hardware–software co-design, 22 on knowledge transfer, and 15 on deployment case studies. This quantitative coverage highlights both the breadth and the focus of the present review.

The search keywords included the following: "AI miniaturization," "edge intelligence," "model compression," "low-power AI," "deployment," and "embedded systems."

The following inclusion and exclusion criteria are applied.

Inclusion criteria were as follows:

(i) publications written in English;
(ii) works closely related to AI miniaturization, deployment architectures, or energy efficiency optimization;
(iii) review papers or high-quality (frequently cited) research articles.

Exclusion criteria were as follows:

(i) non-English publications;
(ii) works without full-text availability;
(iii) papers weakly related to the topic.

The screening workflow is documented in the Preferred Reporting Items for Systematic Reviews and Meta-Analyses (PRISMA) diagram as shown in Figure A1 in Appendix A.1.

1.4. Organization of This Review

Section 2 reviews the developmental trajectory of AI, systematically compares representative computing architectures, and analyzes their differences in latency, energy efficiency, cost, and structural adaptability. Section 3 focuses on the necessity of AI miniaturization and proposes structural design principles and a practice-oriented framework accordingly. Section 4 concludes the paper.

2. Existing Work

2.1. Exponentially Increasing Demands on Software and Hardware Resources for AI

AI is commonly defined as the use of machines to solve problems that would otherwise require human intelligence [1]. This intrinsic association with "replacing human capabilities" has, since its inception, endowed AI with seemingly limitless application potential [27]. As a general-purpose technology [2], AI has indeed achieved remarkable success across numerous domains, and its applications are now expanding to almost all sectors of the economy.

One important indicator of AI's technological progress is the exponential growth in the parameter counts of leading AI models over the past two decades. As shown in Figure 1, from 2003 to 2024 [28], model sizes developed under the leadership of different entities—academia, industry, government institutions, and research consortia—have grown significantly, with industry-led models displaying a particularly accelerated expansion trend. This phenomenon aligns closely with the power–law relationship highlighted by the Scaling Laws for Neural Language Models (2020), which shows an approximate power–law correlation among model size, dataset volume, and performance.

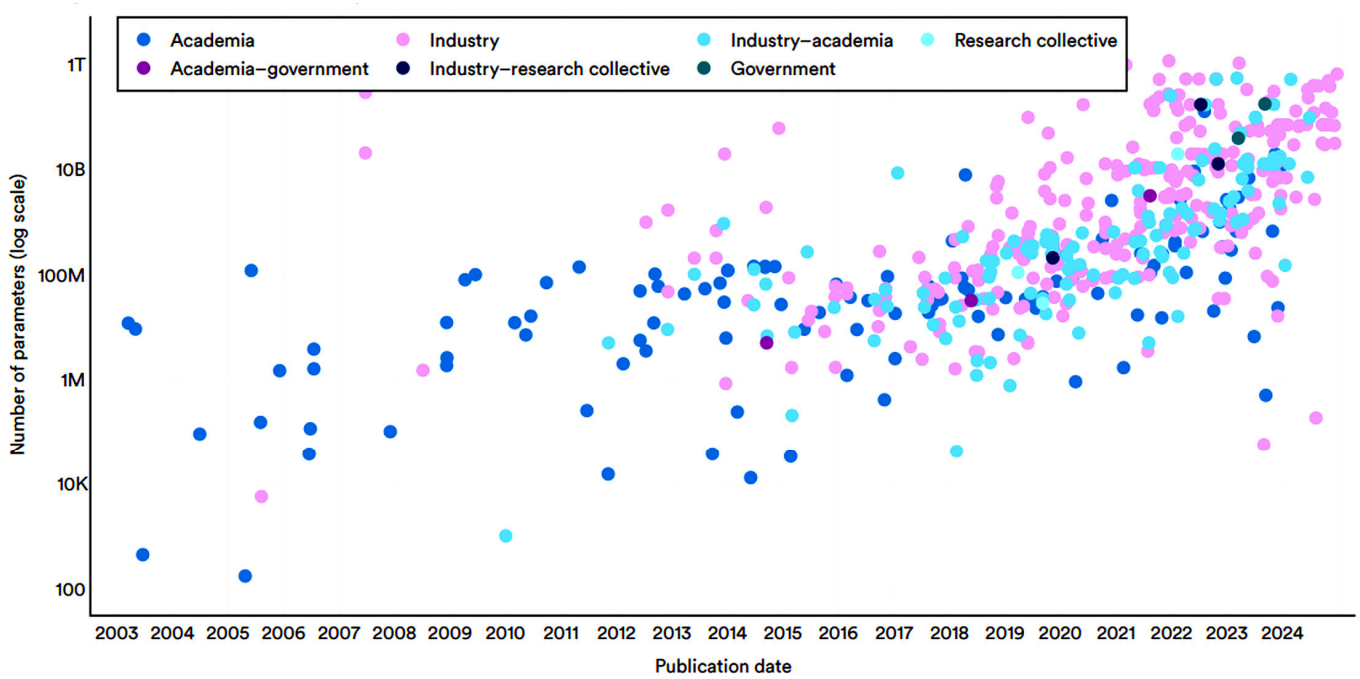

Figure 1. Number of parameters of notable AI models by sector, 2003–2024.

However, AI's remarkable capabilities come at a substantial cost, namely the continuous growth of model size and energy consumption. Performance gains are fundamentally underpinned by ever-increasing demands for computational power [3,10]. Taking two high-impact subfields (natural language processing [4] and computer vision (CV) [6]) as an example, as illustrated in Figure 2, the computation load of AI models has increased significantly over time. Early convolutional networks such as AlexNet [29] and ResNet-50 [4] contained only millions of parameters and required only a few giga floating point operations (GFLOPs) per inference. With the introduction of transformer-based architectures, model sizes and computational costs expanded rapidly, as seen in BERT [7], GPT-2 [30], ViT-L/16 [6], and GPT-3 [31]. When GPT-4 [3] was released, its parameters were not officially disclosed. Therefore, our analysis focuses on compute and cost figures reported in neutral sources [32] rather than speculative parameter estimates.

This exponential expansion highlights the urgency of model miniaturization strategies. When AI models are concerned, the historical growth in graphics processing unit (GPU) requirements for training major systems is summarized in Table 2, which illustrates the rapid increase in computational demands from AlexNet to GPT-4. The table complements Figure 2 by providing the corresponding training hardware and GPU-equivalent estimates, thereby linking parameter scaling with infrastructure requirements.

As further examples, both computation load and computational resource requirements have risen sharply, as quantitatively validated in multiple studies, such as OpenAI's AI and compute tracking reports (2023) and Patterson et al.'s systematic analysis of deep learning energy consumption [10]. At the model-training level, GPU requirements for major AI models have also increased dramatically over time [31]. The trajectory from AlexNet to GPT-4, as summarized in Table 2, clearly illustrates this escalation. The rising GPU usage and compute requirements reported in the table are consistent with the floating point operation (FLOP) scaling trends highlighted by OpenAI's AI and compute reports (2023) and further corroborate the industry's heavy reliance on massive computational resources for training large-scale language models and multimodal systems in recent years.

Taken together, Figure 1 and Table 2 underscore a critical challenge: performance improvements in AI have been achieved at the expense of continuous resource expansion,

resulting in sharply rising energy consumption, hardware costs, and environmental burdens. Consequently, optimizing computational efficiency while maintaining performance has become one of the core scientific challenges in AI development [3,10,33]. This establishes the central contradiction between performance expansion and unsustainable energy and deployment costs and sets the logical starting point for the subsequent discussion on cloud–edge trade-offs and miniaturization strategies.

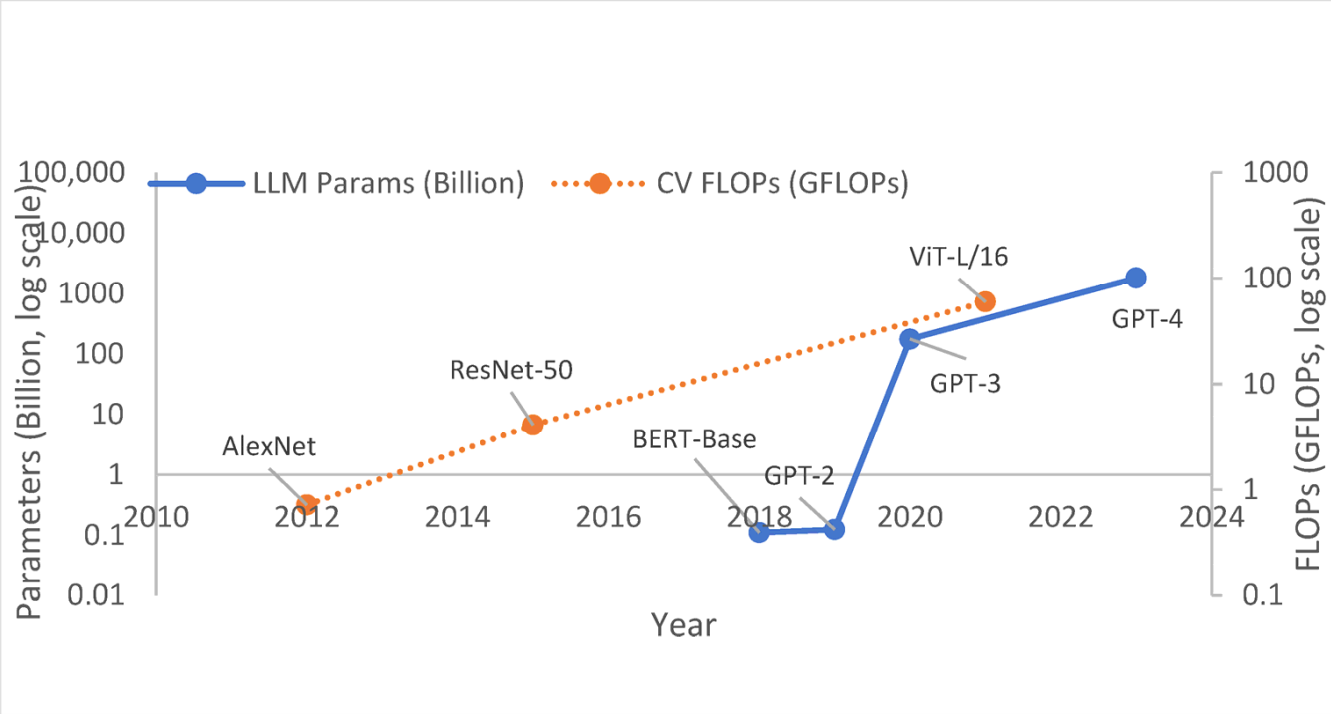

Figure 2. Growth of parameters and compute requirements of notable AI models, 2012–2024. Data sources: Stanford AI Index 2024/2025 and model technical reports [34–40].

Table 2. Training resources of representative AI models, 2012–2024. Sources: Stanford AI Index 2024/2025 and OpenAI technical reports [34–40].

Model Name	Parameter Size (B)	Training Hardware Used	Estimated Equivalent A100 GPUs
AlexNet (2012)	0.06	GTX 580 × 2	≈0.01 A100-equivalent
BERT-Large (2018)	0.34	Tensor Processing Unit v3 × 64	≈4 A100 GPUs
GPT-2 (2019)	1.5	V100 × hundreds	≈50 A100 GPUs
GPT-3 (2020)	175	V100 × 3640	≈2000 A100 GPUs
GPT-4 (2023)	Not disclosed;	A100 × 25,000+	≈25,000 A100 GPUs
GPT-4 Turbo (2024)	Not disclosed; energy-first deployment (OpenAI, 2024)	H100/A100 hybrid deployment	Not disclosed, but energy-first deployment (undisclosed)

2.2. Cloud and Edge Computing Form a Trade-Off Solution for Existing AI Systems

2.2.1. Cloud Computing

The massive computational requirements of modern AI models often exceed the capacity of individuals or small-to-medium enterprises. As a result, cloud computing has become the mainstream platform for AI services, and any meaningful discussion of AI development and deployment must involve cloud computing [34]. With its pay-as-you-go

model and remote accessibility, cloud computing effectively addresses this challenge [35] and serves as the backbone for training and deploying large-scale AI models.

According to application scenarios, cloud computing can be categorized into the following types:

(i) Public Clouds. Platforms such as Amazon Web Services (AWS), Azure, and Google Cloud require no local infrastructure, provide elastic scalability, and adopt a pay-as-you-go model. These platforms form the foundation of large-scale AI deployment, supporting services such as GPT-4 and Google Bard. However, they suffer from limited data privacy [36], latency issues, and reduced customization due to shared infrastructure. This classification and definition were formally provided by National Institute of Standards and Technology (NIST) [41];

(ii) Private Clouds. Represented by IBM Cloud Private and OpenStack, private clouds offer enhanced security and customization, making them suitable for sensitive sectors such as finance, healthcare, and government. Typical applications include medical image analysis and banking risk models. However, they involve high construction and maintenance costs [37];

(iii) Hybrid Clouds. Solutions such as Azure Stack and Amazon Outposts enable flexible switching between local and cloud environments. These architectures are often used in "cloud-based training and on-premises deployment" models, particularly for industrial AI and autonomous driving. Yet, their management complexity remains a major challenge;

(iv) Community Clouds. Typically used in consortia involving healthcare, research, or public institutions, community clouds support resource sharing and cost distribution across multiple organizations. Representative applications include federated diagnostic model training within healthcare alliances or GPU sharing among universities. However, their user scope is limited, and governance issues remain significant [38];

(v) Edge Clouds. Platforms such as NVIDIA Edge Computing Platform (EGX), Huawei Cloud Edge, and edge clouds operated by telecom carriers emphasize local processing to reduce backhaul bandwidth consumption, thereby achieving low latency [12]. This concept has already been incorporated into discussions of edge intelligence [19,42].

Overall, cloud computing provides the computational backbone for large-scale raw data processing and AI model training. However, its strength in absorbing and processing vast heterogeneous datasets also imposes tremendous pressure on centralized infrastructures, particularly when facing increasingly complex and diversified data streams [43]. According to NIST's classical definition, cloud computing is characterized as a model of on-demand self-service, measured service, resource pooling, rapid elasticity, and broad network access [41]. Its proliferation has enabled highly efficient yet energy-intensive support for AI.

2.2.2. Edge Computing

The high centralization of cloud computing limits its ability to support real-time applications, since tasks are executed in remote data centers that must be shared by large numbers of users, with resource allocation often guided by economic efficiency. In this context, edge computing has gradually emerged as a standard approach to alleviating the computational burden on the cloud, particularly for data-intensive and real-time AI applications [26]. Targeting Internet of Things (IoT) and smart device scenarios, edge computing preprocesses raw data locally, thereby reducing transmission latency and cloud-side load [44].

In recent practice, edge computing has evolved into multiple architectural forms designed to optimize performance in specific application scenarios. The four most representative categories include the following:

(i) Fog Computing. Proposed and formalized by Cisco, fog computing emphasizes pushing computation and storage downward to network-layer nodes closer to the data source, thereby achieving lower latency and higher reliability [45];
(ii) Cloudlet Computing. Introduced by Satyanarayanan, this concept involves deploying small-scale virtualized data centers near wireless access points, enabling mobile devices to access low-latency computational support over wireless connections [46];
(iii) Mobile Edge Computing. Standardized by the European Telecommunications Standards Institute (ETSI), MEC features deep integration with telecommunication access networks. Typical applications include vehicular networks, emergency response systems, and the industrial IoT [47] (ETSI MEC White Paper, 2014/2019/2022);
(iv) Micro Data Centers (MDCs). Designed to replicate full data center functionality at the edge, MDCs are suitable for high-investment scenarios such as industrial automation and remote environmental monitoring [42].

Overall, edge computing is not a replacement for cloud computing but rather a strategic complement, especially for latency-critical and privacy-sensitive applications [22]. Research in edge intelligence suggests that only through a rational partitioning of the cloud–edge boundary can system-level trade-offs between performance and energy consumption be achieved [19].

2.3. AI Miniaturization Strategies and Technical Pathways

With the widespread adoption of AI, the demand for computational power has risen sharply, primarily driven by the exponential growth in model size. Kaplan et al. (2020) revealed a power–law relationship among model scale, dataset volume, and performance, explaining why accuracy improvements often require exponential increases in computational resources [33]. Recent surveys indicate that redundancy reduction (e.g., pruning, quantization, and distillation), device-side optimization, and cross-layer compiler/operator-level co-optimization have gradually formed a systematic methodology [23–25,48]. These studies provide the methodological starting point for the three core strategies discussed below.

As shown in Figure 3 [28,32], since 2019, the number of chips used for large-scale training has expanded dramatically, with associated hardware costs and energy consumption also escalating sharply, as documented in the Stanford AI Index 2024/2025. If such trends persist, computational and energy requirements may continue to rise beyond sustainable limits [49]. Together with Figure 2, which depicts the exponential scaling of model parameters and FLOPs, and Table 3, which summarizes the escalating training costs and diminishing marginal gains across frontier AI models, these results highlight that performance improvements achieved by hardware stacking entail unsustainable growth in energy and financial overhead. With power consumption emerging as a hard constraint on further AI development, miniaturization is a necessary and irreversible trend [50].

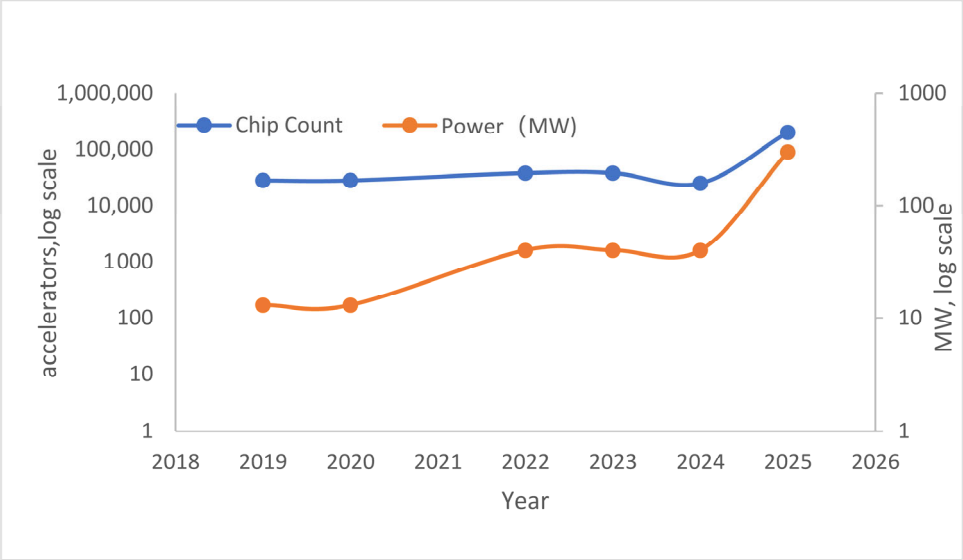

Figure 3. The unchecked expansion of AI systems is unsustainable. This conclusion has been echoed in multiple studies [10].

Table 3. Training costs and performance gaps of frontier AI models, based on data from Stanford AI Index Reports 2024 [3,32].

Year	Model	Training Cost (USD)	log10(Cost)	Metric (Elo Ranking)	Value (%)
2017	Transformer	≈670	2.83	–	–
2019	RoBERTa Large	≈160,000	5.2	–	–
2023	GPT-4	≈79,000,000	7.9	Top-1 vs. Top-10 gap	11.9
2023	GPT-4	≈79,000,000	7.9	Top-1 vs. Top-2 gap	4.9
2024	Llama 3.1-405B	≈170,000,000	8.23	Top-1 vs. Top-2 gap	0.7

In summary, since power consumption has become a hard constraint on AI development [51], miniaturization is not only a necessity but also an irreversible trend.

2.3.1. Major Strategies for AI Miniaturization

The development of AI fundamentally depends on three key elements: models, data [33], and computational resources. Therefore, efforts toward miniaturization must simultaneously target all three dimensions, giving rise to three core strategies: redundancy reduction, knowledge transfer, and hardware–software co-design [52–54].

(1) Redundancy Reduction

The central idea of this strategy is to simplify model structures by removing unnecessary complexity. Several representative techniques include the following:

(i) Model Pruning [52]. By removing unimportant neurons or connections, pruning reduces model size. Han et al. introduced a three-step approach consisting of pruning, quantization, and entropy coding, achieving up to a 49× compression ratio on CNNs. However, pruning remains challenging in large-scale transformer models. Recently, movement pruning was proposed to adaptively select sparsity patterns during fine-tuning based on gradient migration strength, achieving significantly greater sparsity while maintaining a comparable level of accuracy. This method has become representative of adaptive pruning [55];

(ii) Quantization. Reducing the parameter precision from 32 bit to 8 bit or lower can significantly decrease storage and computational overhead [56]. Representative approaches include INT8 quantization combined with quantization-aware training (QAT), which

maintains accuracy while achieving acceleration and compression [56]. More recently, FP8 formats have been adopted for training and inference, reducing bandwidth and memory usage while improving throughput [57]. Even more aggressive 4-bit methods such as SmoothQuant (2023), Activation-aware Weight Quantization (AWQ) [58], and GPTQ [59] further compress parameters while preserving robustness, making the deployment of language models (LLMs) on consumer-grade GPUs feasible;

(iii) Low-Rank Decomposition [60]. This technique approximates weight tensors through matrix factorization, thereby reducing computational complexity. It is commonly applied to compress fully connected layers or attention weights;

(iv) Lightweight Architecture Design [61]. Models such as MobileNet and ShuffleNet, which are based on depthwise separable convolutions or channel shuffling, are specifically designed for mobile deployment.

(2) Knowledge Transfer

(i) Knowledge Distillation [53]. Small models are trained to mimic the outputs of larger models. Hinton et al. first proposed the use of "soft labels" to transfer knowledge effectively;

(ii) Parameter Sharing [62]. Sharing weights across related tasks reduces the total number of parameters, thereby improving efficiency;

(iii) Transfer Learning. Cross-domain applications are enabled by fine-tuning pretrained models or extracting transferable features;

(iv) Parameter-Efficient Fine-Tuning. Recent methods such as Low-Rank Adaptation (LoRA) [63] freeze the base model weights while introducing only low-rank adaptation matrices, significantly reducing the number of trainable parameters. Building on this, Quantized Low-Rank Adaptation (QLoRA) [64] combines four-bit weight quantization with the LoRA paradigm, enabling the efficient fine-tuning of large models even in single-GPU or low-memory environments. These approaches have become critical techniques for adapting large models to resource-constrained platforms.

(3) Hardware–Software Co-Design

(i) Neural Architecture Search [65]. NAS automates the design of model architectures under hardware constraints. For example, EfficientNet employs a compound scaling strategy to balance depth, width, and resolution, achieving both efficiency and accuracy.

(ii) Computation Graph Compilation Optimization [48]. At the framework level, Tensor Virtual Machine (TVM) [48] enables end-to-end operator generation and tuning; Open Neural Network Exchange (ONNX) Runtime [55] and Accelerated Linear Algebra (XLA) [66] accelerate inference through graph fusion and cross-hardware optimization; and NVIDIA TensorRT (TensorRT) further leverages operator-level optimizations on GPUs to deliver low latency and high throughput.

(iii) Deployment-Specific Optimizations [54]. These include caching strategies, batch-size tuning, and operator fusion. More recently, research has proposed KV cache compression and quantization methods. For instance, ZipCache [67] employs saliency-based selection to reduce redundant cache entries, significantly lowering memory footprint while preserving accuracy in long-context inference.

(4) Integration of Strategies.

It should be emphasized that these three categories of strategies are not mutually exclusive but rather interdependent and complementary. They are often combined into practical hybrid optimization pipelines [25,68,69].

2.3.2. Major Directions/Scenarios for AI Miniaturization

The application scenarios of AI miniaturization can be broadly categorized into two deployment priorities: energy-first systems and performance-first systems. Each

emphasizes different optimization strategies under specific constraints, and representative cases have already emerged in practice.

(1) Energy-First Systems

Energy-first systems are typically deployed in resource-constrained environments such as wearable devices, sensor networks, and low-power edge nodes. Their primary objective is to extend system lifetime under limited battery supply.

The following representative optimization strategies are commonly considered.

(i) Early Exit. Terminates inference once a target confidence level is reached to avoid redundant computation [70];
(ii) Cascade Detectors. Use lightweight models for initial screening, followed by more complex models only when necessary [25];
(iii) Adaptive Sampling and Hierarchical Pipelines. Reduce transmission frequency and bandwidth consumption by structuring layered processing [19];
(iv) Dynamic Voltage and Frequency Scaling (DVFS)-aware Scheduling. Dynamically adjusts voltage and frequency to reduce power consumption [42];
(v) Small Context Windows. Restrict context length in language tasks to reduce memory and computational overhead [64].

The following cases are commonly found in energy-first systems.

Case 1: Wearable Perception Agents.

In human activity recognition (HAR) tasks on wearable devices, lightweight neural networks combined with early exit strategies have been shown to reduce inference latency and energy consumption while maintaining high recognition accuracy [71,72].

Case 2: Embedded Vision and IoT Camera.

In embedded vision and IoT camera applications, optimized lightweight real-time detection networks have been proposed to balance speed and accuracy while reducing resource consumption [73].

Case 3: Industrial Material Inspection.

In resource-constrained construction scenarios, lightweight hybrid models have also been explored. For example, an Ultrasonic-AI Hybrid eXtreme Gradient Boosting (XGBoost) approach has been applied for material defect detection, demonstrating the potential of combining traditional machine learning with efficient sensing under limited energy budgets [74].

(2) Performance-First Systems

Performance-first systems are mainly designed for complex perception tasks and large-scale language understanding tasks. Their primary objective is to maintain high accuracy and low latency while achieving the maximal compression of model size.

Representative optimization strategies:

(i) Split Computing. Partitions the model so that the early layers are deployed on the device side, while subsequent layers are executed in the cloud [23];
(ii) Near-End Refinement. Edge devices perform low-precision inference, while cloud servers provide high-precision correction [75];
(iii) Speculative Decoding. Makes parallel predictions of multiple candidate outputs to reduce response latency [76];
(iv) Key–Value (KV) Cache Reuse. Reduces redundant computation in large LLMs during long-context tasks [67];
(v) Micro-Batching. Improves throughput and hardware utilization by running small-batch parallel workloads. While widely adopted in cloud-based GPU scenarios, its application to resource-constrained edge devices requires further exploration.

(vi) Performance-first systems can be found in the following cases.

Case 1: In-Vehicle Perception Systems [77].

In autonomous driving perception tasks, split computing has been applied by deploying feature extraction on the vehicle side while delegating complex inference to edge servers. Empirical studies show that this paradigm reduces end-to-end latency while maintaining high perception accuracy [78].

Case 2: Mobile LLM Inference.

For large language model inference on mobile devices, researchers have proposed combining parameter-efficient fine-tuning (e.g., LoRA) with KV cache reuse to improve inference efficiency and responsiveness. For example, MobiLoRA [79] leverages context-aware cache optimization to significantly improve latency and throughput performance in mobile LLM inference.

Case 3: Infrastructure Monitoring Robots.

In performance-critical construction monitoring, a 3D Vision Crack Robot has been developed to enable real-time crack detection, combining 3D vision technologies with edge inference to illustrate how AI-driven perception can be deployed in robotics under stringent accuracy and latency requirements [80].

Taken together, the application scenarios of AI miniaturization reveal two polarized deployment priorities: energy-first systems and performance-first systems. These scenarios demonstrate that the three core strategies (redundancy reduction, knowledge transfer, and hardware–software co-design) cannot be applied in isolation but must be flexibly combined according to deployment demands. Looking ahead, case-driven research and standardized practices will be critical to enabling the widespread adoption of low-power, high-performance AI. To concretize these insights, Table 4 summarizes how the strategies presented in Section 2.3.1 are manifested in different combinations across these two deployment types.

Table 4. AI miniaturization strategies across deployment scenarios.

Application	Scenario Type	Core Principle	Applied Techniques
Huawei Watch GT Series	Energy-First	Redundancy Compression HW–SW Co-Design	Pruning, Quantization MindSpore Lite Optimization Platform Adaptation
NVIDIA Jetson Nano	Energy-First	Redundancy Compression HW–SW Co-Design	Pruning, Quantization Lightweight Architecture (e.g., MobileNet, ShuffleNet) TensorRT Fusion Deployment Optimization
YOLOv5-Nano/v8-Nano	Performance-First	Redundancy Compression HW–SW Co-Design	NAS Lightweight Architecture Quantization (e.g., INT8, FP16) Custom Mobile Structure Inference Engine Integration
MobileNetV3	Performance-First	Knowledge Transfer Redundancy Transfer	Knowledge Distillation (e.g., DistilBERT, TinyBERT) Parameter Sharing Pruning
DistilBERT/TinyBERT	Performance-First	Knowledge Transfer HW–SW Co-Design	Knowledge Distillation Weight Reduction Structure Simplification
Once-for-All (OFA)	Performance-First	Knowledge Transfer Redundancy Compression	Supernetwork Distillation Submodel Transfer NAS + Multi-Platform Generation

3. Discussion and Recommendations

3.1. AI Miniaturization as a Key Step in AI Development

It is widely recognized that AI and its computational infrastructure have developed rapidly. AI models continue to push performance boundaries, permeating the diverse domains of daily life, work, and industry, giving the impression of entering an "AI for everything" era. Yet behind this apparent prosperity lies a critical reality: the enormous resource consumption of AI. In terms of chip count, computational demand, energy consumption, and model scale, AI development has exhibited exponential growth. This "resource hunger" poses a severe constraint on the sustainable advancement of AI.

Although the path of "unconstrained expansion" has yielded notable performance gains, it has also introduced problems of high energy consumption, high costs, and high deployment barriers. In the future, computational and energy resources cannot increase without bound, and cost constraints may risk reducing AI from a technology that empowers the many to a privilege serving only a few. As a core technology entrusted with driving societal transformation, AI should aim not merely for "peak performance" but for "universal accessibility." This echoes the "productivity paradox" identified by Brynjolfsson et al. [2]: highly capable AI systems have not consistently translated into broad economic productivity, largely due to barriers in deployment, accessibility, and integration.

Against this backdrop, a clearer developmental direction emerges: AI must evolve toward miniaturization, low power consumption, and efficiency. Only in this way can AI overcome excessive dependence on resources, integrate into a wider range of everyday devices and practical scenarios, and become deployable, sustainable, and universally accessible intelligent infrastructure.

3.2. AI Miniaturization: A Natural Evolution Rather than a Mere Compromise

When discussing the future of AI under the hard constraints of computational capacity, cost, and energy, the concept of miniaturization frequently arises. However, AI miniaturization should not be seen merely as a passive response to limitations; rather, it represents an inherent and rational trajectory within the evolution of AI technologies. To regard it solely as a supplementary deployment strategy to large models would underestimate both its technological potential and its long-term significance. In fact, AI miniaturization can be viewed as a natural evolutionary direction, rather than a temporary compromise.

While in high-precision, multi-task, cloud-centric applications, miniaturized models may not match large models in absolute accuracy or overall performance, they demonstrate remarkable adaptability in low-power and real-time edge deployment scenarios. This explains why miniaturized models are often positioned as "fallback options" under constrained conditions. Yet such a perception, though practically reasonable, does not adequately capture their deeper role in the structural evolution of AI.

Fundamentally, AI miniaturization seeks to realize rich intelligent functions within limited resource budgets through optimal system architectures. This involves eliminating redundancy, compressing computation, and controlling energy consumption to achieve structural efficiency. Its foundation lies in understanding the boundaries of AI capability: large models explore the limits of intelligence and sketch the blueprint, while miniaturization translates that blueprint into deployable systems. Together, they form a complementary relationship of "vision and realization."

3.3. Trends in AI Miniaturization: Insights from the Evolution of Embedded Systems

As AI systems evolve toward efficiency, compactness, and ubiquity, the structural trade-off between performance and scale becomes increasingly critical. In this context, the historical evolution of embedded systems offers instructive reference. Embedded systems

have advanced from cabinet-sized hardware to millimeter-scale chips, a trajectory enabled by structural reconfiguration and high levels of integration.

By analogy, AI miniaturization should not be understood merely as compressing models, but as a process of architectural reconfiguration, representational redesign, and computational optimization aimed at improving energy efficiency. This broader approach can ultimately achieve a higher computational density and greater scalability in real-world applications.

3.4. Reflections on AI Miniaturization, Productization, and Implementation Pathways

The functional drivers of AI can be broadly categorized into consumer-driven and production-driven motivations. The release of ChatGPT by OpenAI is widely regarded as a critical turning point that reignited the global wave of AI enthusiasm. Before this, although AI technologies had been applied at length in key domains such as finance, industry, and healthcare, they remained largely "invisible" to the public. ChatGPT, as the first highly interactive and general-purpose application to capture public attention, marked a milestone in the productization of AI. With AI now attracting significant public interest, its role in shaping both the economy and everyday life must be considered, and miniaturization and its implementation pathways are particularly important.

3.4.1. The Expanding Role of AI Through Miniaturization

Although AI has achieved broad public recognition and been applied across multiple fields, in terms of actual deployment structures, most applications still concentrate on production-oriented domains such as industrial chains, supply chains, and infrastructure services. This distribution suggests that while consumer-facing AI products enjoy high visibility and user engagement, they primarily act as accelerators of technological diffusion rather than the core engine of AI democratization.

Drawing from the experience of past industrial revolutions, the transformative power of AI lies not only in technological breakthroughs but also in the deep restructuring of production relations and value chains. From this perspective, AI should be considered as an integral component of a new industrial revolution, with its core value being the reconfiguration of productivity, rather than mere sensationalism at the level of perception.

Accordingly, the conceptual pathway of AI miniaturization should not be confined to superficial goals such as "shrinking physical size" or "embedded deployment," nor should it merely cater to the limited computational capacities of consumer devices. The recent consumer demand for miniaturized models is, to a large extent, a reactive response to the large-model wave triggered by ChatGPT—essentially, the idea that "even low-end devices must appear intelligent." Such thinking is overly shaped by short-term economic incentives, while overlooking the fundamental value of AI.

A more constructive pathway for AI miniaturization should focus on enhancing productivity while also ensuring deployability. From this standpoint, revisiting the developmental trajectory of embedded systems is particularly instructive. The miniaturization of embedded systems was not simply a physical downsizing of general-purpose computers; rather, it entailed systematic optimization of tasks, structures, and energy efficiency. This process enabled autonomous decision-making and enhanced system responsiveness at the device level. Such historical success provides crucial structural insights for the future of AI miniaturization.

3.4.2. Not Just Making AI Smaller, but Making AI-Driven Devices Lighter

For AI systems, adaptability to the target environment is more important than merely shrinking their size. The essence of miniaturization should not be understood as "making AI systems smaller," but as "reducing the execution burden on end devices." Consequently,

the design of AI systems should consistently account for task complexity, power budgets, and inference pathways, shifting the design logic from "Can it run?" to "Can it run efficiently and economically?"

The evolution of embedded systems reinforces this logic: their goal was never to forcibly embed the full functionality of PCs, but rather to restructure tasks so that devices could effectively assume them. This represents precisely the kind of embedded mindset that AI miniaturization should draw upon: one centered on task definition, streamlining, standardization, and hardware–software co-design, rather than assumptions of unlimited resources.

3.4.3. Co-Evolution of AI Chips and Models: Computation as an Endogenous Structural Resource

In real-time applications with low to moderate computational complexity, AI should be regarded as an endogenous component of device architecture rather than an external add-on module. This requires the co-evolution of model structures and hardware architectures, forming a co-design workflow.

Drawing from the co-development paradigm of embedded systems, AI miniaturization should facilitate the migration of decision-making capabilities into the functional units of devices. Instead of relying solely on a centralized AI system (e.g., cloud computing), lightweight task-specific submodels can be distributed across device layers to enhance responsiveness and deployment flexibility.

3.4.4. Ecological Symbiosis of AI Models: Integrating Local Intelligence with Global Optimization

Future AI deployments will increasingly exhibit structural decentralization, multi-node collaboration, and high real-time responsiveness. This trend points to an "ecological symbiosis" architecture: each device integrates localized intelligent units or algorithms tailored to its task requirements, enabling autonomous decision-making and collaborative sensing. This shift parallels the evolution of embedded systems from purely logical control to localized intelligent processing.

At the same time, centralized AI should transition from being the sole decision-maker to serving as a coordinator and capability orchestrator, balancing local anomaly handling with global optimization. Such a paradigm not only enhances system resilience and resource efficiency but also alleviates the burden on centralized infrastructures.

3.5. Methodological Framework and Implementation Pathways

The preceding discussion indicates that AI miniaturization is not only a natural trajectory of technological development but also exhibits differentiated demands across application scenarios, typically categorized into energy-first and performance-first systems. However, how to translate these technical pathways into actionable engineering processes remains insufficiently systematized.

To address this gap, and building upon the methodological synthesis presented in Section 2, this review proposes a general four-stage framework:

(i) Profiling (Performance and Energy Characterization). Establishes quantitative profiles of performance, latency, energy consumption, and memory footprint;
(ii) Partitioning (Computation and Communication Allocation). Divides computational tasks and communication loads across cloud, edge, and device layers according to system constraints;
(iii) Optimization (Constraint-Driven Combinatorial Tuning). Applies joint optimization under multi-dimensional constraints, balancing accuracy, latency, energy efficiency, and memory usage;

(iv) Validation (Standardized Verification). Conducts standardized evaluation and benchmarking to ensure replicability, reliability, and fairness across deployment environments.

This framework aims to guarantee accuracy while simultaneously meeting constraints on latency, energy, and memory, thereby providing a reusable implementation pathway for AI miniaturization across diverse deployment scenarios. To provide a clearer view, the overall workflow of the proposed framework is illustrated in Figure 4, where the four stages are complemented by feedback loops and deliverables.

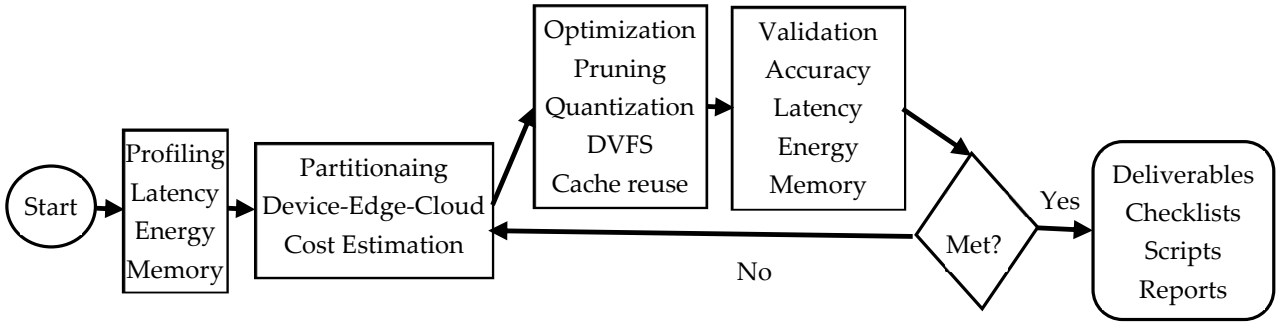

Figure 4. Practical design framework for AI miniaturization.

3.5.1. Profiling (Performance and Energy Characterization)

In Figure 4, the first step is to profile the performance and energy characteristics of the existing model in order to identify the primary bottlenecks in system execution. Key metrics to be measured include the following:

(i) Computational Overhead. The computational load and inference latency of individual layers or operators;
(ii) Memory Footprint. Peak memory usage of model weights and intermediate activations;
(iii) Communication Performance. Uplink and downlink bandwidth, as well as round-trip latency;
(iv) Energy Distribution. Power consumption and runtime of different modules.

The total energy consumption can be expressed as follows:

$$E_{\text{total}} = \sum_{i} P_i t_i \tag{1}$$

where P_i denotes the power consumption of module i, and t_i is its execution time.

The outcome of this step is a bottleneck inventory and an intuitive performance profile, providing decision support for subsequent stages.

3.5.2. Partitioning (Computation and Communication Allocation)

The second step is to determine how computation and data transmission should be partitioned across devices, edge nodes, and the cloud.

Common strategies include the following:

(i) Device–Cloud Split Computing. Partitioning model layers between local devices and the cloud;
(ii) Cascade/Two-Stage Filtering. Lightweight local models perform preliminary screening before invoking more complex remote models;
(iii) Edge-Side Refinement. Performing low-precision inference at the edge, with cloud servers providing high-precision correction;
(iv) Streaming or Batch Transmission. Selecting transmission modes depending on bandwidth and latency constraints.

The end-to-end latency can be expressed as follows:

$$t_{\text{total}} = t_{\text{device}} + t_{\text{network}} + t_{\text{cloud}} \quad (2)$$

where t_{device} denotes the processing latency at the device side, t_{network} the transmission delay, and t_{cloud} the processing latency at the cloud side.

If the actual latency or energy consumption exceeds predefined constraints, the partitioning must be reconsidered or passed to the optimization stage.

3.5.3. Optimization (Method Mapping and Combination)

The third step is to map the identified bottlenecks to appropriate optimization methods. Common approaches can be grouped into the following categories:

(i) Latency Reduction. Compiler and runtime optimizations [81], operator fusion, speculative decoding, and parallel or pipelined execution;
(ii) Energy Reduction. Early exiting, adaptive sampling and hierarchical pipelines, DVFS scheduling, and compressed or pruned data transmission;
(iii) Memory Footprint Reduction. Low-bit quantization (INT8, FP8, four-bit), cache compression or quantization, and activation checkpointing;
(iv) Model Size Reduction. Structured or movement pruning, knowledge distillation, low-rank decomposition, and parameter-efficient fine-tuning methods such as LoRA/QLoRA;
(v) Collaborative Optimization. Device–cloud partitioning, near-end refinement, and cache reuse.

The guiding principle during optimization is to prioritize lossless or near-lossless techniques; methods with minor accuracy degradation can be considered next; and only as a last resort should techniques involving noticeable performance trade-offs be applied in exchange for gains in energy efficiency or latency.

3.5.4. Validation (Verification and Regression)

The final step is to conduct a standardized validation of the optimized system to ensure that the solution is reliable and practical under real-world operating conditions.

Key evaluation metrics typically include the following:

(i) Model Accuracy. Metrics such as classification accuracy and detection precision;
(ii) Latency. Not only mean latency but also tail latency measures such as P95;
(iii) Energy Consumption per Inference. The energy required for an inference pass;
(iv) Peak Memory Usage. Maximum memory footprint during execution;
(v) Stability. Indicators such as thermal behavior, throttling events, and error rates;
(vi) Cost. Both hardware acquisition and operational expenses.

If the results fail to meet the predefined requirements, the process should revert to the previous step for adjustment. It is recommended that the entire workflow be implemented as standardized scripts and reporting templates, enabling repeated execution under varying hardware and network conditions to ensure reproducibility and comparability.

3.6. Design Checklist

The proposed methodological framework provides a complete workflow from analysis to optimization. However, for practical deployment, a concise design checklist is also required to guide decisions under different objectives. To this end, we further distill the framework into a comparative summary of the differentiated requirements of energy-first and performance-first systems.

As shown in Table 5, the checklist covers objectives, constraints, method selection, trade-off outcomes, and validation, serving as a practical reference tool for system deployment and evaluation.

Table 5. Design checklist for AI miniaturization under energy- and performance-first priorities.

Dimension	Energy-First Systems	Performance-First Systems
Objective	Maximize battery lifetime; reduce overall energy consumption	Minimize end-to-end latency; increase throughput
Constraints	Limited memory capacity and strict energy budget	Accuracy floor must be satisfied; higher hardware or network cost acceptable
Methods	Early exit (e.g., multi-branch networks), cascaded detectors, adaptive sampling, DVFS-based scheduling	Split computing, near-end refinement, speculative decoding, cache reuse
Trade-offs	Energy saving and longer device lifetime, but possible minor accuracy or latency degradation	Lower latency and higher throughput, but increased power consumption and deployment cost
Validation	Focus on power profiling, runtime stability, and reproducibility across hardware	Focus on latency distribution, throughput, and system scalability across workloads

This checklist not only contrasts the objectives and constraints of energy-first versus performance-first systems but also clarifies the trade-offs resulting from different method selections. By directly linking design decisions with validation standards, the checklist functions as a "bridge" between the strategic review in Section 2 and the methodological framework in Section 3, thereby offering more actionable guidance for practical deployment.

4. Conclusions

This review provided a systematic review of the key bottlenecks in AI related to energy consumption, computational demand, and deployment efficiency. It summarized three core strategies for AI miniaturization: redundancy reduction, knowledge transfer, and hardware–software co-design. Based on the differentiated requirements of energy-first and performance-first application scenarios, this review proposed a four-stage methodological framework (profiling → partitioning → optimization → validation) together with a design checklist. These tools offer a unified and process-oriented approach for system design under varying objectives.

From a developmental perspective, AI miniaturization is shifting from isolated point-level optimizations toward cross-layer integration: pruning, quantization, and distillation at the model level continue to enhance deployability, while split computing, speculative decoding, and runtime optimization at the system level significantly improve end-to-end performance. Compared with existing surveys, the distinctive contribution of this work lies not only in synthesizing methodological strategies but also in introducing framework- and checklist-based tools that translate fragmented optimization techniques into reusable engineering pathways.

Future research should further expand cross-scenario empirical validation and standardized benchmarking, while reinforcing hardware–software co-design, in order to advance AI systems toward greater efficiency, sustainability, and accessibility. Although the number of case studies included here remains limited, the proposed framework and checklist provide actionable tools for subsequent research and practice, helping to address the gap left by prior surveys that have often focused on a single dimension.

Author Contributions: Conceptualization, B.T. and S.D.; methodology, B.T. and S.D.; validation, B.T., S.D. and A.J.S.; formal analysis, B.T.; investigation, B.T.; resources, B.T.; data curation, B.T.; writing—original draft preparation, B.T.; writing—review and editing, B.T., S.D. and A.J.S.; visualization, B.T., S.D. and A.J.S.; supervision, S.D. and A.J.S.. All authors have read and agreed to the published version of the manuscript.

Funding: This study is supported in part by the National Research Foundation (NRF) of South Africa (SA), Grant Numbers SRUG2203291049 and EQP240515218724.

Data Availability Statement: No new data were created or analyzed in this study.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AI	Artificial Intelligence
AWQ	Activation-aware Weight Quantization
AWS	Amazon Web Services
CV	Computer Vision
DVFS	Dynamic Voltage and Frequency Scaling
EGX	NVIDIA Edge Computing Platform
ETSI	European Telecommunications Standards Institute
FLOPs	Floating-Point Operations
FP16	16-bit Floating Point
FP8	8-bit Floating Point
GFLOPs	Giga Floating-Point Operations
GPU	Graphics Processing Unit
GPT	Generative Pre-trained Transformer
HAR	Human Activity Recognition
INT8	8-bit Integer
IoT	Internet of Things
KV Cache	Key–Value Cache
LLM	Large Language Model
LoRA	Low-Rank Adaptation
QLoRA	Quantized Low-Rank Adaptation
MDC	Micro Data Center
MEC	Mobile Edge Computing
NAS	Neural Architecture Search
NIST	National Institute of Standards and Technology
NLP	Natural Language Processing
OFA	Once-for-All
ONNX	Open Neural Network Exchange
PRISMA	Preferred Reporting Items for Systematic Reviews and Meta-Analyses
QAT	Quantization-Aware Training
TensorRT	NVIDIA TensorRT
TVM	Tensor Virtual Machine
XGBoost	eXtreme Gradient Boosting
XLA	Accelerated Linear Algebra
YOLO	You Only Look Once

Appendix A

Appendix A.1. PRISMA Flow Diagram

The PRISMA flow diagram shows the systematic screening and selection process in full detail.

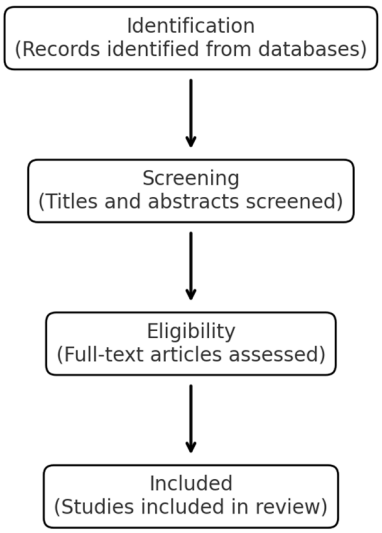

Figure A1. PRISMA flow diagram illustrating the literature identification, screening, eligibility assessment, and final inclusion process of this review.

Appendix A.2. Data and Normalization Disclosure

This appendix consolidates the data sources, normalization methods, and limitations for all figures and tables (Figures 1–3; Tables 2–4) as well as representative case studies cited in this review.

(1) Data Sources

Figure 1 (Model parameters by actor, 2003–2024): Data obtained from Stanford AI Index 2024/2025 and OpenAI AI and Compute reports. Models were classified as Academia, Industry, or Collaboration based on the primary affiliation of the lead institution reported in the AI Index. Specifically, models led by universities or research institutes were categorized as Academia; those led by companies as Industry; and joint efforts as Collaboration. Only milestone models with parameter counts verifiable from neutral sources were included. GPT-4 data are excluded from Figures 2 and 3 due to lack of disclosure.

Figure 2 (Model parameters and compute, 2012–2024): Representative models with references available in the bibliography were retained as follows.

AlexNet (2012): Krizhevsky [29]

GPT-2 (2019): OpenAI GPT-2 technical report.

GPT-3 (2020): Brown et al. [31]

Other milestones (e.g., ResNet-50, BERT, ViT, GPT-4) were excluded to maintain consistency with the reference list.

Table 2 (Training hardware and GPU-equivalents): Hardware counts taken from Stanford AI Index 2024/2025 and model technical reports. GPU-equivalents (A100-equivalent) are reported only when provided by neutral sources; no speculative conversions were introduced.

Figure 3 (Hardware scaling and energy/cost trends, 2019–2025): Compiled from Stanford AI Index 2024/2025 and neutral analyses [10]. Reported values are indicative trends, not precise year-by-year measurements.

Table 3 (Training costs and performance gaps): Costs derived from Stanford AI Index 2024/2025 (R&D section). Performance metrics (Top-1 vs. Top-k differences) taken from AI Index benchmark datasets.

Table 4 (Strategy–Scenario summary): Examples (Huawei Watch GT, Jetson Nano, You Only Look Oncev5-Nano (YOLO), MobileNetV3, DistilBERT, Once-for-All) collected from published papers, official documentation, and review surveys.

(2) Definitions and Normalization

Parameters (B): number of trainable parameters in billions.

Compute (FLOPs): training floating-point operations FLOPs; measured when available, otherwise estimated.

Estimation formula (for transformers):

$$\text{FLOPs} = 2 \times N \times T,\ T = 200 \tag{A1}$$

where N is parameter size (in billions) and T is sequence length (default 200 tokens).

CNN FLOPs: AlexNet, GPT-2, and GPT-3 FLOPs taken as measured inference FLOPs at standard input size.

GPU-equivalent (A100-eq): normalization applied only if provided by neutral sources.

Training cost (USD): reported expenditure at time of publication; no inflation adjustment.

Energy (E): Energy figures, when mentioned, are quoted directly from neutral reports without any site-level normalization, PUE-based adjustment, or secondary computation.

(3) Estimation vs. Reported Values

Figure 2: FLOPs are estimated for transformers using Equation (A1) but measured for AlexNet.

Table 2: Hardware counts were reported by the literature.

Figure 3: Presents trend indicators only, not tied to individual model training runs.

Table 3: Costs and performance gaps reported directly by Stanford AI Index, without secondary estimation.

(4) Uncertainty and Caveats

Undisclosed data (e.g., GPT-4 parameters) are excluded from quantitative comparison and are noted only in background discussions.

Community estimates: not included in figures/tables and cited separately as context.

Comparability limits: efficiency varies with datasets, optimizers, and precision. The results are directional instead of forensic.

Mixed hardware deployments: Retained in original form (e.g., A100/H100 hybrid) without conversion.

Revision risk: Later reports may update costs or chip counts, and this appendix reflects sources available up to June 2025.

References

1. McCarthy, J.; Minsky, M.L.; Rochester, N.; Shannon, C.E. A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence. *AI Mag.* **2006**, *27*, 12–14. [CrossRef]
2. Brynjolfsson, E.; Rock, D.; Syverson, C. *Artificial Intelligence and the Modern Productivity Paradox: A Clash of Expectations and Statistics*; National Bureau of Economic Research: Cambridge, MA, USA, 2017; w24001.
3. Achiam, J.; Adler, S.; Agarwal, S.; Ahmad, L.; Akkaya, I.; Aleman, F.L.; Almeida, D.; Altenschmidt, J.; Altman, S.; et al.; OpenAI. GPT-4 Technical Report. *arXiv* **2023**, arXiv:2303.08774. [CrossRef]
4. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. *Image Recognit.* **2015**, *7*, 327–336.
5. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 779–788.
6. Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. An Image Is Worth 16 × 16 Words: Transformers for Image Recognition at Scale. *arXiv* **2021**, arXiv:2010.11929.
7. Devlin, J.; Chang, M.-W.; Lee, K.; Toutanova, K. BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Minneapolis, MN, USA, 2–7 June 2019; pp. 4171–4186.

8. Radford, A.; Narasimhan, K.; Salimans, T.; Sutskever, I. *Improving Language Understanding by Generative Pre-Training*; OpenAI Technical Report; 2018. Available online: https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf (accessed on 11 October 2025).
9. Jumper, J.; Evans, R.; Pritzel, A.; Green, T.; Figurnov, M.; Ronneberger, O.; Tunyasuvunakool, K.; Bates, R.; Žídek, A.; Potapenko, A.; et al. Highly Accurate Protein Structure Prediction with AlphaFold. *Nature* **2021**, *596*, 583–589. [CrossRef]
10. Strubell, E.; Ganesh, A.; McCallum, A. Energy and Policy Considerations for Deep Learning in NLP. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, Florence, Italy, 28 July–2 August 2019; pp. 3645–3650.
11. Gonzales, J.T. Implications of AI Innovation on Economic Growth: A Panel Data Study. *J. Econ. Struct.* **2023**, *12*, 13. [CrossRef]
12. Bender, E.M.; Gebru, T.; McMillan-Major, A.; Shmitchell, S. On the Dangers of Stochastic Parrots: Can Language Models Be Too Big? In Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency, Toronto, ON, Canada, NY, USA, 3–10 March 2021; ACM: New York, NY, USA; pp. 610–623.
13. Vinuesa, R.; Azizpour, H.; Leite, I.; Balaam, M.; Dignum, V.; Domisch, S.; Felländer, A.; Langhans, S.D.; Tegmark, M.; Fuso Nerini, F. The Role of Artificial Intelligence in Achieving the Sustainable Development Goals. *Nat. Commun.* **2020**, *11*, 233. [CrossRef]
14. OECD. *Measuring the Environmental Impacts of Artificial Intelligence Compute and Applications: The AI Footprint*; OECD Digital Economy Papers. No. 341; OECD Publishing: Paris, France, 2022. [CrossRef]
15. Sinha, S.; State of IoT 2024: Number of Connected IoT Devices Growing 13% to 18.8 Billion Globally. IoT Anal. 2024. Available online: https://iot-analytics.com/number-connected-iot-devices/ (accessed on 9 October 2025).
16. Hammi, B.; Khatoun, R.; Zeadally, S.; Fayad, A.; Khoukhi, L. IoT Technologiesfor Smart Cities. *IET Netw.* **2018**, *7*, 1–13. [CrossRef]
17. Gubbi, J.; Buyya, R.; Marusic, S.; Palaniswami, M. Internet of Things (IoT): A Vision, Architectural Elements, and Future Directions. *Future Gener. Comput. Syst.* **2013**, *29*, 1645–1660. [CrossRef]
18. Shi, W.; Cao, J.; Zhang, Q.; Li, Y.; Xu, L. Edge Computing: Vision and Challenges. *IEEE Internet Things J.* **2016**, *3*, 637–646. [CrossRef]
19. Zhou, Z.; Chen, X.; Li, E.; Zeng, L.; Luo, K.; Zhang, J. Edge Intelligence: Paving the Last Mile of Artificial Intelligence with Edge Computing. *Proc. IEEE* **2019**, *107*, 1738–1762. [CrossRef]
20. Mosenia, A.; Jha, N.K. A Comprehensive Study of Security of Internet-of-Things. *IEEE Trans. Emerg. Top. Comput.* **2017**, *5*, 586–602. [CrossRef]
21. Li, H.; Ota, K.; Dong, M. Learning IoT in Edge: Deep Learning for the Internet of Things with Edge Computing. *IEEE Netw.* **2018**, *32*, 96–101. [CrossRef]
22. Duan, Q.; Huang, J.; Hu, S.; Deng, R.; Lu, Z.; Yu, S. Combining Federated Learning and Edge Computing Toward Ubiquitous Intelligence in 6G Network: Challenges, Recent Advances, and Future Directions. *IEEE Commun. Surv. Tutor.* **2023**, *25*, 2892–2950. [CrossRef]
23. Wang, X.; Tang, Z.; Guo, J.; Meng, T.; Wang, C.; Wang, T.; Jia, W. Empowering Edge Intelligence: A Comprehensive Survey on On-Device AI Models. *ACM Comput. Surv.* **2025**, *57*, 1–39. [CrossRef]
24. Dantas, P.V.; Sabino Da Silva, W.; Cordeiro, L.C.; Carvalho, C.B. A Comprehensive Review of Model Compression Techniques in Machine Learning. *Appl. Intell.* **2024**, *54*, 11804–11844. [CrossRef]
25. Liu, D.; Zhu, Y.; Liu, Z.; Liu, Y.; Han, C.; Tian, J.; Li, R.; Yi, W. A Survey of Model Compression Techniques: Past, Present, and Future. *Front. Robot. AI* **2025**, *12*, 1518965. [CrossRef]
26. Deng, S.; Zhao, H.; Fang, W.; Yin, J.; Dustdar, S.; Zomaya, A.Y. Edge Intelligence: The Confluence of Edge Computing and Artificial Intelligence. *IEEE Internet Things J.* **2020**, *7*, 7457–7469. [CrossRef]
27. Chéour, R.; Jmal, M.W.; Khriji, S.; El Houssaini, D.; Trigona, C.; Abid, M.; Kanoun, O. Towards Hybrid Energy-Efficient Power Management in Wireless Sensor Networks. *Sensors* **2021**, *22*, 301. [CrossRef]
28. Maslej, N. Artificial Intelligence Index Report 2025. *arXiv* **2025**, arXiv:2504.07139. [CrossRef]
29. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet Classification with Deep Convolutional Neural Networks. In Proceedings of the Advances in Neural Information Processing Systems, Lake Tahoe, NV, USA, 3–6 December 2012; Curran Associates, Inc.: New York, NY, USA, 2012; Volume 25.
30. Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; Sutskever, I. *Language Models Are Unsupervised Multitask Learners*; OpenAI Technical Report; OpenAI: San Francisco, CA, USA, 2019; Available online: https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf (accessed on 11 October 2025).
31. Brown, T.B.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. Language Models Are Few-Shot Learners. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 1877–1901.
32. The 2025 AI Index Report | Stanford HAI. Available online: https://hai.stanford.edu/ai-index/2025-ai-index-report (accessed on 24 September 2025).
33. Kaplan, J.; McCandlish, S.; Henighan, T.; Brown, T.B.; Chess, B.; Child, R.; Gray, S.; Radford, A.; Wu, J.; Amodei, D. Scaling Laws for Neural Language Models. *arXiv* **2020**, arXiv:2001.08361. [CrossRef]

34. Van Der Vlist, F.; Helmond, A.; Ferrari, F. Big AI: Cloud Infrastructure Dependence and the Industrialisation of Artificial Intelligence. *Big Data Soc.* **2024**, *11*, 20539517241232630. [CrossRef]
35. Zhang, Q.; Cheng, L.; Boutaba, R. Cloud Computing: State-of-the-Art and Research Challenges. *J. Internet Serv. Appl.* **2010**, *1*, 7–18. [CrossRef]
36. Sikeridis, D.; Papapanagiotou, I.; Rimal, B.P.; Devetsikiotis, M. A Comparative Taxonomy and Survey of Public Cloud Infrastructure Vendors. *Comput. Netw.* **2020**, arXiv:1710.01476*168*, 107019.
37. Srivastava, P.; Khan, R. A Review Paper on Cloud Computing. *Int. J. Adv. Res. Comput. Sci. Softw. Eng.* **2017**, *7*, 362–365. [CrossRef]
38. Goyal, S. Public vs Private vs. Hybrid vs. Community—Cloud Computing: A Critical Review. *Int. J. Comput. Netw. Inf. Secur.* **2014**, *6*, 20–29. [CrossRef]
39. Moghaddam, S.M.S.H.; Dashtdar, M.; Jafari, H. AI Applications in Smart Cities' Energy Systems Automation. *Repa Proceeding Ser.* **2022**, *3*, 1–5. [CrossRef]
40. Sun, L.; Jiang, X.; Ren, H.; Guo, Y. Edge-Cloud Computing and Artificial Intelligence in Internet of Medical Things: Architecture, Technology and Application. *IEEE Access* **2020**, *8*, 101079–101092. [CrossRef]
41. Mell, P.; Grance, T. *The NIST Definition of Cloud Computing*; National Institute of Standards and Technology (NIST): Gaithersburg, MD, USA, 2011; p. 7.
42. Iftikhar, S.; Gill, S.S.; Song, C.; Xu, M.; Aslanpour, M.S.; Toosi, A.N.; Du, J.; Wu, H.; Ghosh, S.; Chowdhury, D.; et al. AI-Based Fog and Edge Computing: A Systematic Review, Taxonomy and Future Directions. *Internet Things* **2023**, *21*, 100674. [CrossRef]
43. Admin, L.; Elmirghani, J. A Survey of Big Data Machine Learning Applications Optimization in Cloud Data Centers and Networks. *Appl. Sci.* **2021**, *11*, 7291. [CrossRef]
44. Khan, W.Z.; Ahmed, E.; Hakak, S.; Yaqoob, I.; Ahmed, A. Edge Computing: A Survey. *Future Gener. Comput. Syst.* **2019**, *97*, 219–235. [CrossRef]
45. Bonomi, F.; Milito, R.; Zhu, J.; Addepalli, S. Fog Computing and Its Role in the Internet of Things. In Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing, Helsinki, Finland, 17 August 2012; ACM: New York, NY, USA, 2012; pp. 13–16.
46. Satyanarayanan, M.; Bahl, P.; Caceres, R.; Davies, N. The Case for VM-Based Cloudlets in Mobile Computing. *IEEE Pervasive Comput.* **2009**, *8*, 14–23. [CrossRef]
47. Welcome to the World of Standards! Available online: https://www.etsi.org/ (accessed on 26 September 2025).
48. Chen, T.; Moreau, T.; Jiang, Z.; Zheng, L.; Yan, E.; Cowan, M.; Shen, H.; Wang, L.; Hu, Y.; Ceze, L.; et al. TVM: An Automated End-to-End Optimizing Compiler for Deep Learning. In Proceedings of the 13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18), Carlsbad, CA, USA, 8–10 October 2018; USENIX Association: Berkeley, CA, USA, 2018; pp. 578–594.
49. Pilz, K.F.; Sanders, J.; Rahman, R.; Heim, L. Trends in AI Supercomputers. *arXiv* **2025**, arXiv:2504.16026. [CrossRef]
50. Mishra, A.; Nurvitadhi, E.; Cook, J.J.; Marr, D. Wrpn: Wide Reduced-Precision Networks. *arXiv* **2018**, arXiv:1709.01134.
51. Stiefel, K.S.; Coggan, J. The Energy Challenges of Artificial Superintelligence. *Front. Artif. Intell.* **2022**, *5*, 1–5. [CrossRef]
52. Han, S.; Mao, H.; Dally, W.J. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. *arXiv* **2016**, arXiv:1510.00149. [CrossRef]
53. Hinton, G.; Vinyals, O.; Dean, J. Distilling the Knowledge in a Neural Network. *arXiv* **2015**, arXiv:1503.02531. [CrossRef]
54. Ren, A.; Zhang, T.; Ye, S.; Li, J.; Xu, W.; Qian, X.; Lin, X.; Wang, Y. ADMM-NN: An Algorithm-Hardware Co-Design Framework of DNNs Using Alternating Direction Methods of Multipliers. In Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems, Providence, RI, USA, 13–17 April 2019; ACM: New York, NY, USA, 2019; pp. 925–938.
55. Sanh, V.; Wolf, T.; Rush, A.M. Movement Pruning: Adaptive Sparsity by Fine-Tuning. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 20378–20389.
56. Jacob, B.; Kligys, S.; Chen, B.; Zhu, M.; Tang, M.; Howard, A.; Adam, H.; Kalenichenko, D. Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 2704–2713.
57. Micikevicius, P.; Stosic, D.; Burgess, N.; Cornea, M.; Dubey, P.; Grisenthwaite, R.; Ha, S.; Heinecke, A.; Judd, P.; Kamalu, J.; et al. FP8 Formats for Deep Learning. *arXiv* **2022**, arXiv:2209.05433. [CrossRef]
58. Lin, J.; Tang, J.; Tang, H.; Yang, S.; Xiao, G.; Han, S. AWQ: Activation-Aware Weight Quantization for On-Device LLM Compression and Acceleration. *GetMobile Mob. Comput. Commun.* **2025**, *28*, 12–17. [CrossRef]
59. Frantar, E.; Ashkboos, S.; Hoefler, T.; Alistarh, D. GPTQ: Accurate Post-Training Quantization for Generative Pre-Trained Transformers. *arXiv* **2023**, arXiv:2210.17323.
60. Denton, R.; Zaremba, W.; Bruna, J.; LeCun, Y.; Fergus, R. Exploiting Linear Structure Within Convolutional Networks for Efficient Evaluation. *Adv. Neural Inf. Process. Syst.* **2014**, *27*, 1269–1277.

61. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv* **2017**, arXiv:1704.04861. [CrossRef]
62. Lan, Z.; Chen, M.; Goodman, S.; Gimpel, K.; Sharma, P.; Soricut, R. ALBERT: A Lite BERT for Self-Supervised Learning of Language Representations. *arXiv* **2020**, arXiv:1909.11942. [CrossRef]
63. Hu, E.J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; Chen, W. LoRA: Low-Rank Adaptation of Large Language Models. *ICLR* **2022**, *1*, 3.
64. Dettmers, T.; Pagnoni, A.; Holtzman, A.; Zettlemoyer, L. QLORA: Efficient Finetuning of Quantized LLMs. *Adv. Neural Inf. Process. Syst.* **2023**, *36*, 10088–10115.
65. Tan, M.; Le, Q.V. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. *arXiv* **2019**, arXiv:1905.11946.
66. Snider, D.; Liang, R. Operator Fusion in XLA: Analysis and Evaluation. *arXiv* **2023**, arXiv:2301.13062. [CrossRef]
67. He, Y.; Zhang, L.; Wu, W.; Liu, J.; Zhou, H.; Zhuang, B. ZipCache: Accurate and Efficient KV Cache Quantization with Salient Token Identification. In Proceedings of the 38th Conference on Neural Information Processing Systems, Vancouver, BC, Canada, 10–15 December 2024; Curran Associates, Inc.: New Orleans, LA, USA, 2024; Volume 37. forthcoming.
68. Li, M.; Liu, Y.; Liu, X.; Sun, Q.; You, X.; Yang, H.; Luan, Z.; Gan, L.; Yang, G.; Qian, D. The Deep Learning Compiler: A Comprehensive Survey. *IEEE Trans. Parallel Distrib. Syst.* **2021**, *32*, 708–727. [CrossRef]
69. Zhu, X.; Li, J.; Liu, Y.; Ma, C.; Wang, W. A Survey on Model Compression for Large Language Models. *Trans. Assoc. Comput. Linguist.* **2023**, *12*, 1556–1577. [CrossRef]
70. Jazbec, M.; Timans, A.; Veljkovic, T.H.; Sakmann, K.; Zhang, D.; Naesseth, C.A.; Nalisnick, E. Fast yet Safe: Early-Exiting with Risk Control. *Adv. Neural Inf. Process. Syst.* **2024**, *37*, 129825–129854.
71. Lattanzi, E.; Contoli, C.; Freschi, V. Do We Need Early Exit Networks in Human Activity Recognition? *Eng. Appl. Artif. Intell.* **2023**, *121*, 106035. [CrossRef]
72. Yu, J.; Zhang, L.; Cheng, D.; Huang, W.; Wu, H.; Song, A. Improving Human Activity Recognition With Wearable Sensors Through BEE: Leveraging Early Exit and Gradient Boosting. *IEEE Trans. Neural Syst. Rehabil. Eng.* **2024**, *32*, 3452–3464. [CrossRef] [PubMed]
73. Chen, R.; Wang, P.; Lin, B.; Wang, L.; Zeng, X.; Hu, X.; Yuan, J.; Li, J.; Ren, J.; Zhao, H. An Optimized Lightweight Real-Time Detection Network Model for IoT Embedded Devices. *Sci. Rep.* **2025**, *15*, 3839. [CrossRef]
74. Wan, S.; Li, S.; Chen, Z.; Tang, Y. An Ultrasonic-AI Hybrid Approach for Predicting Void Defects in Concrete-Filled Steel Tubes via Enhanced XGBoost with Bayesian Optimization. *Case Stud. Constr. Mater.* **2025**, *22*, e04359. [CrossRef]
75. Singh, R.; Gill, S.S. Edge AI: A Survey. *Internet Things Cyber-Phys. Syst.* **2023**, *3*, 71–92. [CrossRef]
76. Xu, D.; Yin, W.; Zhang, H.; Jin, X.; Zhang, Y.; Wei, S.; Xu, M.; Liu, X. EdgeLLM: Fast On-Device LLM Inference With Speculative Decoding. *IEEE Trans. Mob. Comput.* **2025**, *24*, 3256–3273. [CrossRef]
77. Chen, L.; Feng, D.; Feng, E.; Zhao, R.; Wang, Y.; Xia, Y.; Chen, H.; Xu, P. HeteroLLM: Accelerating Large Language Model Inference on Mobile SoCs Platform with Heterogeneous AI Accelerators. *arXiv* **2025**, arXiv:2501.14794.
78. Ulicny, C.; Rauch, R.; Gazda, J.; Becvar, Z. Split Computing in Autonomous Mobility for Efficient Semantic Segmentation Using Transformers. In Proceedings of the 2025 IEEE Symposia on Computational Intelligence for Energy, Transport and Environmental Sustainability (CIETES), Trondheim, Norway, 17–20 March 2025; IEEE: Piscataway, NJ, USA, 2025; pp. 1–8.
79. Li, B.; Wang, Y.; Ma, H.; Chen, L.; Xiao, J.; Wang, S. MobiLoRA: Accelerating LoRA-Based LLM Inference on Mobile Devices via Context-Aware KV Cache Optimization. In Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Vienna, Austria, 27 July–1 August 2025; Association for Computational Linguistics: Stroudsburg, PA, USA, 2025; pp. 23400–23410.
80. Hu, K.; Chen, Z.; Kang, H.; Tang, Y. 3D Vision Technologies for a Self-Developed Structural External Crack Damage Recognition Robot. *Autom. Constr.* **2024**, *159*, 105262. [CrossRef]
81. Xia, C.; Zhao, J.; Sun, Q.; Wang, Z.; Wen, Y.; Yu, T.; Feng, X.; Cui, H. Optimizing Deep Learning Inference via Global Analysis and Tensor Expressions. In Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, La Jolla, CA, USA, 27 April–1 May 2024; ACM: New York, NY, USA, 2024; Volume 1, pp. 286–301.