

Neural Network-Oriented Chip Architecture Design for Edge AI: Co-Optimization of Efficient Algorithms and Low-Power Hardware

Yuchuan Yue

*School of Engineering, University of Manchester, Manchester, UK
yuchuan.yue@student.manchester.ac.uk*

Abstract. Edge Artificial Intelligence (Edge AI) is increasingly used to deploy deep learning models on embedded devices. It reduces latency and improves privacy compared with cloud computing. However, edge devices are constrained by limited computation, memory, and power resources. As a result, efficient neural network deployment becomes a key challenge. This paper examines how to design neural network-oriented chip architectures under such constraints. In particular, it focuses on the co-optimization of efficient algorithms and low-power hardware. It analyzes the computational characteristics of neural networks and the constraints of edge devices. Meanwhile, it reviews typical accelerator architectures, including processing element arrays, dataflow design, and memory hierarchy. In addition, it discusses algorithm-level optimization methods such as quantization and pruning, as well as hardware-level low-power techniques. The results show that hardware-software co-design is necessary to achieve high performance and energy efficiency, and point to future research directions in Edge AI chip design.

Keywords: Edge AI, Neural Network Accelerator, Hardware-Software Co-design, Low-Power Design, Dataflow Optimization

1. Introduction

Fueled by recent advances, Artificial Intelligence (AI) has developed rapidly, with deep neural networks achieving significant success in applications such as computer vision, natural language processing, and speech recognition. Traditionally, these models are deployed in cloud-based systems, where large computational and memory resources are available. However, the growing demand for real-time processing, low latency, and data privacy has driven the development of Edge AI, which enables neural network inference directly on embedded and mobile devices. Despite its advantages, Edge AI systems are subject to strict constraints in computational capability, memory capacity, and power consumption. These limitations make it difficult to efficiently deploy complex neural network models on resource-constrained devices. While existing studies have explored various optimization techniques, they often focus on individual design levels and may not fully address the interaction between computation, memory access, and energy efficiency under edge constraints. To address these challenges, this paper explores neural network-oriented chip

architecture design for Edge AI systems, with a focus on hardware-software co-design. By jointly optimizing algorithm, architecture, and circuit levels, it aims to boost performance and energy efficiency while satisfying resource constraints. In particular, this paper investigates the computational characteristics of neural networks and the constraints of Edge AI systems. It also reviews key architectural components, including processing element arrays, dataflow organization, and memory hierarchy. Moreover, it discusses algorithm-level optimization and low-power hardware design techniques, while also presenting cross-layer co-design strategies and future research directions.

2. Key features of Edge AI and neural network computing

2.1. Design constraints in Edge AI systems

Edge AI systems operate on resource-constrained platforms such as smartphones, IoT sensors, and embedded devices, where constraints in power, memory, and computational resources fundamentally govern neural network accelerator design. In particular, power is constrained by battery and thermal limits. Energy-efficient architectures are required to reduce switching activity and minimize data movement, which are often major sources of energy consumption in edge inference workloads [1]. Besides, memory capacity and bandwidth also impose significant restrictions. Deep neural networks typically require substantial storage and frequent memory access, while edge devices are constrained by limited on-chip memory and costly off-chip transfers. Existing studies show that data movement can exceed computation in energy cost, thus making memory hierarchy optimization and data reuse strategies essential for efficient accelerator design [2]. In addition, real-time processing requirements in applications like autonomous driving and video surveillance demand low latency and predictable execution. This motivates parallel architectures and optimized dataflow for fast and deterministic inference [1]. However, these constraints are tightly coupled and introduce inherent design trade-offs. For example, increasing parallelism may improve throughput but also increases power and memory pressure, requiring joint optimization of performance, energy efficiency, and resource use.

2.2. Computational characteristics of neural networks

Neural networks exhibit characteristics that directly shape accelerator design. First, neural network computation is dominated by matrix operations. More specifically, convolution and fully connected layers can be transformed into general matrix multiplications (GEMM). These operations require a large number of multiply-accumulate (MAC) computations, which form the core workload of deep learning models [3]. As a result, accelerator architectures are designed with parallel compute units, such as processing element (PE) arrays, to efficiently handle structured matrix operations. Second, neural networks exhibit a high degree of parallelism, as neurons within the same layer operate independently, enabling data-parallel execution [3]. This characteristic supports the use of large-scale parallel architectures. However, increasing parallelism raises communication and synchronization costs, requiring careful interconnect and workload mapping in accelerator design. Third, data reuse is an important feature in neural network workloads. Input features and weights are reused multiple times, especially in convolutional layers [1]. Efficient hardware architectures exploit this property through optimized dataflow and buffer design. By improving data reuse, accelerators can significantly reduce memory access and energy consumption. Finally, neural networks are memory-intensive. Large volumes of data must be transferred between memory and compute units,

often becoming the main system bottleneck. Data movement can even consume more energy than computation itself, thus making memory hierarchy design critical for accelerator performance [2].

2.3. Impact of neural network workloads on chip architecture

The constraints of Edge AI systems and neural network characteristics jointly shape chip architecture design. Limited power, memory, and latency restrict hardware resources, while matrix-dominated computation and data reuse define workload behavior. As a result, neural network accelerators must balance computation and data movement efficiently under these constraints [1]. PE arrays provide the core computational capability of neural network accelerators. Since workloads are dominated by matrix operations, accelerators leverage large-scale parallel PE arrays to efficiently perform MAC operations [3]. And architectures such as systolic arrays enable structured data movement and high data locality. However, increasing PE count raises power and communication costs, thus requiring a balance between parallelism and efficiency. Second, dataflow organization determines how data is reused and transferred within the accelerator. Because data movement is costly, efficient dataflow design is essential, with common approaches like weight-, output-, and row-stationary methods that boost data reuse and reduce memory access [2, 4]. Third, memory hierarchy mitigates the bandwidth bottleneck in neural network workloads. Frequent access to large data volumes generates heavy off-chip traffic, so multi-level on-chip buffers and caches are used to improve locality and reduce energy consumption [1]. Finally, domain-specific architectures (DSAs) integrate these design principles into unified hardware systems tailored for neural network workloads, enabling coordinated optimization of computation, data movement, and memory efficiency under resource constraints [1].

3. Neural network-oriented chip architecture design and optimization

3.1. High-level architecture of neural network accelerators

Neural network accelerators can be categorized into several architectural types, each reflecting trade-offs among performance, flexibility, and energy efficiency. Under Edge AI constraints, these trade-offs directly influence hardware selection. In comparison, GPUs provide strong parallel computation capability for neural networks and are well suited for matrix operations and data-parallel workloads. However, as general-purpose processors, they typically incur high power consumption, making them less suitable for resource-constrained edge devices despite their strong performance [1]. Studies also show that GPU-based solutions face limitations in energy efficiency and scalability in edge scenarios [1]. In contrast, TPUs and ASIC-based accelerators are designed specifically for neural network workloads. These domain-specific architectures integrate optimized compute structures, like systolic arrays, to improve both performance and energy efficiency. By reducing control overhead and data movement, they achieve higher performance per watt than GPUs, making them more suitable for Edge AI with strict power and latency constraints [5]. Meanwhile, FPGAs provide reconfigurable hardware acceleration with strong adaptability to different models and real-time requirements. They can achieve lower latency and better energy efficiency than GPUs in certain edge scenarios, but often require complex design effort and generally cannot match the peak performance of ASIC-based solutions [1, 6]. Moreover, DSAs represent the dominant trend in neural network accelerator design, as they tailor computation patterns and dataflows to neural workloads, thus achieving higher energy efficiency and performance under tight resource constraints than general-purpose processors [1].

3.2. Processing element arrays and dataflow organization

Processing element (PE) arrays provide parallel computation capability, while dataflow organization determines how data is reused and moved within the accelerator. Since data movement dominates energy consumption, efficient dataflow design is critical for Edge AI systems [2]. At the compute level, PE arrays consist of multiple simple units performing MAC operations in parallel, enabling high throughput for matrix-based neural network workloads [3]. However, increasing PE count also raises communication and power costs, requiring a balance between parallelism and data movement. On the data management side, dataflow organization defines how inputs, weights, and partial sums are mapped onto PE arrays, aiming to maximize data reuse and reduce memory access [2]. Based on these mappings, common dataflow strategies include weight-stationary (WS), output-stationary (OS), and row-stationary (RS). In WS, weights are kept in processing elements to maximize reuse but increase input and partial-sum movement. In OS, partial sums are stored locally to reduce write operations but require continuous data streaming. In RS, reuse is balanced across weights, inputs, and partial sums, reducing overall data movement at the cost of higher buffer and control complexity.

3.3. Memory hierarchy and data reuse strategies

Memory hierarchy design directly affects the performance and energy efficiency of neural network accelerators. As neural network workloads require intensive data access, bandwidth, latency, and energy efficiency become key considerations in memory system design [2]. Modern accelerators use multi-level memory hierarchies composed of registers, on-chip buffers, and off-chip DRAM. In this hierarchy, on-chip memory provides low-latency and energy-efficient access, while off-chip DRAM provides larger capacity at higher access cost [7]. As a result, reducing off-chip memory access is critical. To reduce memory traffic, neural network accelerators exploit data reuse. Neural networks repeatedly reuse input features, weights, and intermediate results, especially in convolutional layers [1]. Consequently, data reuse reduces movement between memory levels, which is a major source of energy consumption [2]. To improve reuse efficiency, existing strategies mainly enhance data locality. For example, loop tiling and data blocking divide computations into sub-blocks stored in on-chip memory, thereby reducing redundant off-chip access and improving bandwidth utilization. Similarly, buffer allocation stores frequently accessed data near processing elements, which reduces latency and energy cost [3]. Besides, some methods exploit reuse across layers and execution stages. Specifically, cross-layer reuse removes redundant inter-layer transfers, while memory-aware scheduling optimizes data placement and execution order under hardware constraints. Although these methods improve efficiency, they also increase control complexity and hardware overhead [8].

3.4. Cross-layer and hardware-efficient acceleration

Hardware efficiency in Edge AI depends on coordinated optimization across algorithm, circuit, and system levels to reduce computation, memory traffic, and power consumption [9, 10]. At the algorithm level, efficiency is improved by reducing computational redundancy and numerical cost. Quantization replaces high-precision floating-point operations with low-bit representations, significantly reducing arithmetic complexity and memory footprint [9]. Pruning removes redundant connections, producing sparser computation and lower execution cost [9]. Sparsity further skips operations associated with zero-valued activations and weights when supported by hardware. In addition, model compression methods such as weight sharing and knowledge distillation reduce

storage and data transfer overhead while maintaining model performance. At the hardware level, energy efficiency relies on controlling switching activity and power states. For example, clock gating reduces dynamic power by disabling inactive logic during idle cycles. Similarly, DVFS adjusts voltage and frequency based on workload demand to balance energy and performance [11]. In addition, power gating lowers leakage power by shutting down unused circuit blocks, making it suitable for intermittent edge workloads [11]. Besides, approximate computing further improves efficiency by relaxing arithmetic precision requirements and exploiting neural network error tolerance [10]. At the system level, cross-layer co-design jointly optimizes algorithms, architectures, and circuits to reduce inefficiencies across abstraction layers [10]. Quantization and pruning influence hardware utilization patterns, while processing element arrays and memory hierarchies determine data movement efficiency. At the same time, DVFS and clock gating provide additional runtime energy optimization. By aligning algorithm behavior with hardware execution mechanisms, cross-layer co-design reduces overall system overhead rather than isolated local costs.

4. Hardware foundations and design trade-offs for Edge AI accelerators

4.1. CMOS fundamentals for AI hardware

As the physical implementation platform of neural network accelerators, CMOS technology enables dense digital computation with scalable integration and controlled power consumption in modern AI chips. From a performance perspective, CMOS scaling enables higher operating frequency and faster switching speed. This allows accelerators to execute large numbers of MAC operations efficiently. As a result, the high-throughput computation required by neural network workloads is supported by CMOS technology [12]. From a power perspective, CMOS design determines both dynamic and static energy consumption. Dynamic power is related to switching activity, while leakage power becomes significant in advanced technology nodes. Efficient CMOS design enables techniques such as voltage scaling and clock control, which are critical for reducing energy consumption in Edge AI systems [11]. From an integration perspective, CMOS scaling enables high-density integration of processing elements and memory units on a single chip, facilitating parallel architectures such as PE arrays and on-chip memory structures, and resulting in shorter communication distance, higher performance, and better energy efficiency.

4.2. Logic design for neural network modules

Logic design forms the functional basis of neural network accelerators, enabling data processing and control under the resource and latency constraints of Edge AI systems. It must therefore support efficient computation while operating within limited hardware resources and real-time requirements. MAC units perform multiply-accumulate operations and serve as the core computational elements in neural network accelerators. They are designed for high computational density to enable large-scale parallel processing in matrix-based workloads under constrained hardware budgets [13]. In addition, combinational logic implements arithmetic operations such as addition and multiplication without relying on storage elements, enabling fast data processing. In Edge AI systems, it is essential for low-latency execution in real-time inference tasks. Meanwhile, sequential logic, implemented via registers and flip-flops, controls data flow and system timing. And it maintains system state and synchronizes operations across processing elements, ensuring stable and predictable execution under strict timing constraints. Besides, control logic manages the coordination between computation units and memory subsystems by regulating data movement and operation scheduling within the

accelerator. In Edge AI systems, it must efficiently handle dynamic workloads and limited resources while maintaining correct and flexible operation.

4.3. Power, delay, and area trade-offs

Power, delay, and area (PPA) are key metrics for evaluating neural network accelerator designs. In Edge AI systems, these metrics are tightly constrained and must be jointly optimized to satisfy application requirements [11]. Power consumption consists of dynamic and static components, driven by switching activity and leakage effects. Delay determines computational latency, while area reflects hardware resource usage, limiting integration density and scalability [12]. These metrics are strongly coupled and inherently involve trade-offs. Increased parallelism reduces delay but increases area and power consumption. Lowering supply voltage reduces power but may increase delay and degrade performance. Likewise, compact designs reduce area but may constrain throughput, so optimizing one metric typically impacts the others. In Edge AI applications, these trade-offs become more critical due to strict constraints on energy, latency, and chip size. Designers must balance PPA according to application priorities, such as energy efficiency for battery-powered devices or low latency for real-time inference.

5. Future directions for neural network chip architecture

5.1. Hardware trade-offs in Edge AI chips

Edge AI chip design must balance conflicting constraints under limited power, memory, and latency budgets. Thus, performance, efficiency, flexibility, and accuracy cannot be optimized simultaneously. These trade-offs are driven by the strong interdependence among computational capability, energy efficiency, hardware complexity, and model accuracy. In particular, increasing parallelism enhances performance, but this improvement comes at the expense of higher power consumption and increased silicon area. Conversely, power reduction leads to limited computational speed, while architectural flexibility improvement enables broader workload support but results in reduced efficiency. Besides,

higher model accuracy demands substantially greater computation and memory resources. Given these inherent trade-offs, Edge AI chip design typically follows an application-driven paradigm that prioritizes key objectives such as energy efficiency or latency. Accordingly, achieving optimal system performance requires joint optimization across the algorithm, architecture, and circuit levels.

5.2. Model-hardware gaps in Edge AI systems

The proposed design approach is mainly based on conceptual analysis, which may differ from practical deployment in real Edge AI systems. And this discrepancy underscores the limitations of theoretical optimization when applied to real-world hardware environments. Conceptual studies often focus on ideal conditions, where computation, memory access, and system behavior are simplified. Although such models are valuable for understanding design principles, they often fail to capture practical constraints, including hardware variability, system integration complexity, and workload diversity. In contrast, real-world Edge AI deployment involves additional challenges. Hardware platforms must operate under strict power budgets, limited memory capacity, and dynamic workloads. System-level factors, such as communication overhead and runtime variability, can significantly affect actual performance. As a result, theoretical improvements may not directly

translate into practical efficiency gains. These differences introduce challenges in validating and implementing optimized designs. Bridging the gap between conceptual analysis and real deployment requires more comprehensive evaluation and system-level consideration.

5.3. Efficiency and scalability of future designs

Future neural network chip architectures will prioritize efficiency and scalability to meet increasing Edge AI workload complexity under strict resource constraints. One important direction is improving energy efficiency and computational performance through more advanced co-design techniques. By further integrating algorithm optimization with hardware architecture, future accelerators can better reduce computation overhead and data movement. This will enable more efficient execution of deep learning models on edge devices with limited power budgets. Another key direction is enhancing scalability and adaptability. Future architectures must support a wider range of neural network models and dynamic workloads. This requires flexible accelerator architectures combined with optimized memory hierarchies and dataflow mechanisms to maintain performance across different application scenarios. In addition, heterogeneous computing and adaptive hardware mechanisms provide further optimization potential. And such designs enable runtime adaptation of computation and resources to workload characteristics, improving utilization under dynamic conditions.

6. Conclusion

This paper has reviewed and analysed the design of neural network-oriented chip architectures for Edge AI, with a focus on the co-optimization of efficient algorithms and low-power hardware. The analysis indicates that high-performance Edge AI systems rely on coordinated optimization across algorithm, architecture, and circuit levels. In this context, specialized accelerators such as GPUs, ASICs, and NPU are widely used to improve computational efficiency and energy efficiency. Key architectural techniques, including processing element arrays, optimized dataflows, and hierarchical memory systems, help to address the computation and data movement challenges of deep learning workloads. In addition, hardware-friendly algorithm optimizations such as quantization, pruning, and model compression significantly reduce computational complexity and energy consumption. At the circuit level, low-power techniques further enhance overall system efficiency. Besides, cross-layer co-design aligns algorithm behavior with hardware execution characteristics. However, this study is limited to a literature-based review without experimental validation. Future research may explore more integrated design methodologies and emerging paradigms such as processing-in-memory and neuromorphic computing to further improve efficiency.

References

- [1] Sze, V., Chen, Y., Yang, T., & Emer, J. S. (2017). Efficient processing of deep neural networks: A tutorial and survey. *Proceedings of the IEEE*, 105(12), 2295–2329.
- [2] Horowitz, M. (2014). Computing's energy problem (and what we can do about it). In *IEEE ISSCC*.
- [3] Chen, Y., Krishna, T., Emer, J. S., & Sze, V. (2017). Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks. *IEEE Journal of Solid-State Circuits*, 52(1), 127–138.
- [4] Li, S., Chen, X., & Li, J. (2021). Hardware acceleration of deep neural networks: A survey. *ACM Computing Surveys*, 53(6).
- [5] Jouppi, N. P., Young, C., Patil, N., et al. (2017). In-datacenter performance analysis of a tensor processing unit. In *Proceedings of ISCA*.

- [6] Mittal, S. (2020). A survey of FPGA-based accelerators for convolutional neural networks. *Neural Computing and Applications*.
- [7] Chen, T., Du, Z., Sun, N., et al. (2014). DianNao: A small-footprint high-throughput accelerator for ubiquitous machine learning. In *ASPLOS*.
- [8] Samajdar, A., et al. (2018). A systematic methodology for characterizing scalability of DNN accelerators. In *HPCA*.
- [9] Han, S., Mao, H., & Dally, W. J. (2016). Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding. In *International Conference on Learning Representations (ICLR)*.
- [10] Singh, R., & Gill, S. S. (2023). Edge AI: A survey. *Internet of Things and Cyber-Physical Systems*, 3, 71–92.
- [11] Chandrakasan, A., Sheng, S., & Brodersen, R. (1992). Low-power CMOS digital design. *IEEE Journal of Solid-State Circuits*.
- [12] Rabaey, J., Chandrakasan, A., & Nikolic, B. (2003). *Digital integrated circuits: A design perspective*. Prentice Hall.
- [13] Weste, N., & Harris, D. (2011). *CMOS VLSI design: A circuits and systems perspective*. Pearson.