International Journal of
Science, Architecture, Technology and Environment

Evolution of Dedicated Processor Architectures for Generative AI Models (LLMs): A Comprehensive Review

Zainab H. Mohammad[1*]

[1]*Department of Mathematics, Najaf Center, The Open Educational College, Ministry of Education, Najaf, Iraq*

[*]*Correlsponding author: Zainab H. Mohammad, zainabaldulame@gmail.com*

DOI: https://doi.org/10.63680/ijsate0726039.46

Abstract

The scaling of auto-regressive Large Language Models (LLMs) has revolutionized the generative artificial intelligence, and has demonstrated bottlenecks in the architectural design of the traditional von Neumann computing fabrics. The runtime performance ceases to be computation-bound in pre-fill phases, and becomes memory-bandwidth-bound in the iterative token decoding process, with simple models with hundreds of billions of parameters. This incompatibility is manifested in the form of structural memory stalls, underutilization of the execution units and extreme thermal dissipation concerns and all of which are also termed the memory wall. This paper gives a comprehensive survey of the development of dedicated hardware accelerators which are specifically developed to reduce these data transport penalties. We create a systematic taxonomy of Application-Specific Integrated Circuits (ASICs), Reconfigurable Field-Programmable Gate Arrays (FPGAs), and non-von Neumann Processing-In-Memory (PIM) macrostructures. In addition, we discuss the Hardware-Software Co-design paradigm, tracing the direct remodeling of the underlying physical silicon dataflows by low-precision algorithmic tye-castings (INT4/FP4) as well as structural sparsity metrics (2:4). Finally, we also draw attention to the research gaps, i.e., to microarchitectural side-channel vulnerabilities and the extreme heat flux of 3D-stacked semiconductor boundaries, as a clean blueprint to future secure, energy-efficient domain-specific computing cores.

Keywords: Hardware accelerators; Dedicated Processors; Memory Wall; Large Language Models (LLMs); Processing-In-Memory (PIM); Systolic Arrays; Edge AI; Hardware-Software Co-Design; Semiconductor Security.

1. Introduction

The critical structural change of the computational paradigm of artificial intelligence has been driven by the advent of auto-regressive LLMs. Although the core architectures are based on unified multi-head attention models to operate on dense semantic sequence [1], the hardware fabric underpinning these models are facing critical operational performance constraints. A computing environment is no longer limited mainly by the raw arithmetic processing capability, but is now severely limited by the actual physical routing of data, a fact well known as the von Neumann "memory wall" [2]. The huge parameter footprint of frontier models

requires sustained high-bandwidth access to off-chip storage arrays, which poses a significant imbalance between computational and memory data transfer rate [3].

This architectural imbalance poses a two-sided engineering issue when deploying the model. With regard to the initial "pre-fill" step of inference, computation is dense and matrix, which uses conventional spatial parallel structures effectively. The following autoregressive "decoding" step however, scales token by token, turning the workload into a pure memory-bandwidth-limited routine [4]. This system bus needs to stream a continuous stream of huge weight matrices and a growing Key-Value (KV) cache map of each generated token, causing both underutilization of execution units and energy dissipation per token [5]. The semiconductor industry is abandoning homogeneous general-purpose processing structures to support the scaling curves of generative AI without causing a serious power crisis [6-8]. The hardware structures now need to be designed to be maximally energy efficient and process density, as well as specifically to the unique dataflows of transformer components.

2. Taxonomy of Dedicated Processor Architectures

In order to approach the data transport and storage constraints of large-scale neural workloads systematically, microarchitectural design has divided into 3 specialized hardware domains. The domains are each a specific way to set up memory hierarchy, data routing channels, and flexibility of the execution arrays [9], [10].

2.1. Application-Specific Integrated Circuits (ASICs)

The silicon architecture purely customized for dense tensor execution including custom Tensor Processing Units (TPUs), provide the best theoretical profiles of performance-per-watt by eliminating legacy general-purpose x86/ARM control structures [11], [12]. The mode of operation in these processors is mainly based on the 2D spatial systolic arrays, where data flows continuously through an interconnected mesh of localized register files and multiply-accumulate (MAC) cells without having to access global memory banks constantly [13].

This architecture structure significantly reduces the energy penalty of register-files reads and writes. Modern AI ASICs integrate these dense arithmetic arrays with purpose-built hardware logic units to implement non-linear transformer sub-loops, like SwiGLU activations and root-mean-square normalization (RMSNorm), directly in the execution path [14-17]. Nevertheless, the rigidity in the structure of ASICs is an engineering risk in itself; a fundamental change in the algorithms used in the foundational models can make hardwired silicon blocks useless after manufacturing [18-20].

2.2. Reconfigurable FPGA Accelerators

Field-Programmable Gate Arrays (FPGAs) fill the gap between fixed silicon ASICs and general-purpose architectures with a hardware fabric that is runtime-modifiable [21]. FPGAs enable the engineering team to dynamically configure the memory channels and data path bit-widths to the requirements of the changing software, like the shifting KV cache footprints or variable sequence token lengths [22].

With high-level synthesis (HLS) and custom bitstream scheduling, an FPGA can directly map custom matrix dataflows to its internal block RAMs (BRAMs) and digital signal processing (DSP) slices [23], [24]. This custom routing layout removes execution stalls due to fixed register configurations. At much lower maximum clock rates than custom ASICs, FPGAs are important in agility of structure, which is needed to develop generative AI pipelines.

2.3. The silicon architecture purely customized

Processing-In-Memory architectures are one radical departure from conventional von Neumann paradigm in that arithmetic operations are directly done in the memory macros [31]. PIM designs eliminate the energy cost and latency penalty of moving data between high-capacitance system buses by integrating analog or digital logic components into the peripheral circuits of high-density storage arrays [32].

In memory bound LLM decoding layers, crossbar arrays of non-volatile resistive RAM (ReRAM) or phase-change memory (PCM) execute matrix-vector multiplications (MVM) as analog summations of currents using the laws of Kirchhoff and Ohm [33], [34]. This form of execution enables a full weight tensor to get stored and operated upon in the memory block and completely avoids any traditional caching structures [35]. The main microarchitectural challenges to PIM systems include how to manage with non-idealities of analog devices, circuit noise, and large silicon area overhead of fast analog-to-digital converters (ADCs).

3. Literature Review and Comparative Benchmarking

To measure engineering progress of modern dedicated processor architectures to LLM workloads, it is important to compare them against established frameworks and baseline hardware topologies. Conventional general-purpose designs are based on homogenous spatial dataflows and homogenous scheduling that distinguish memory elements and arithmetic execution units [25]. Nonetheless, new peer-reviewed studies have presented dynamic alternative models that optimize hardware resource allocation, data accuracy paths, and temporal scheduling boundaries [26], [27].

3.1. The silicon architecture purely customized

The first attempts at accelerating deep learning models consisted of software pruning, or static quantization maps (FP16 to INT8), on older multi-core graphics platforms [28]. Such methods were temporary solutions in scaling, but they demonstrated severe hardware inefficiencies in the loops of autoregressive generation of frontier transformers [29]. The dynamic token tracking can be unpredictable and the Key-Value (KV) cache may be enormous enough to cause structural memory stalls and significantly reduce the rate of utilization of standard parallel execution arrays [30]. To handle these issues, modern methods consider the concept of mixed-signal computing, processing-in-memory (PIM) topologies, and multi-head self-attention optimization systems of balancing the throughput requirements with strict hardware energy constraints [31], [32].

3.2 Quantitative Comparative Framework

The next in-depth evaluation matrix will entail a rigorous architectural comparison between different processing methodologies, processing engines, and data optimization areas that are reported in recent high impact literature as Table 1:

4. Hardware-Software Co-Design and Algorithmic Compression Accelerations

The microarchitectural scaling limits regarding post-CMOS hardware has shown that physical silicon optimization in isolation is not sufficient to enable frontier generative models. To achieve the maximum efficiency of the execution of LLMs, a Hardware-Software Co-design paradigm is required. In this model, the structural adjustments of neural network topologies are developed concurrently with the underlying compute units, and structural optimizations are inherently run on the hardware pipelines [41].

Table 1: Caption of the table

Study & Reference	Core Methodology / Architectural Approach	Targeted Computational Workload	Principal Performance Advantages	Primary Technical Bottlenecks & Hardware Limits
Jouppi et al. [11], [25]	2D Spatial Systolic Array Architecture (ASIC/TPU)	Dense Matrix-Multiplication ($GEMM$) and Transformer Layers	Maximizes computation-per-watt via localized, register-to-register token data routing.	Rigid execution scheduling; has extremely poor array underutilization with dynamic, variable-length inputs.
Kachris [21], [26]	Reconfigurable FPGA Dataflow Pipelines with Custom Routing	Dynamic Key-Value (KV) Cache Management and Online Inferencing	High operational flexibility; allows post-fabrication bitstream updates to match evolving software rules.	Reduce maximum clock frequencies; bound by parasitic routing capacitance and high reconfigurable logic.
Verma et al. [31], [27]	Non-von Neumann Mixed-Signal Processing-In-Memory (PIM)	Memory-bound Vector-Matrix Processing and Autoregressive Sub-loops	Eliminates system bus data migration penalties by executing logic inside memory macro cell layers.	High sensitivity to analog device noise; requires substantial silicon area for multi-channel ADC/DAC converters.
Xiao et al. [42], [28]	Activation-Aware Mixed-Precision Quantization Engines (INT4/FP4)	Compressed Low-Precision Large Language Model Inference	Reduces global off-chip memory bandwidth requirements by up to 4\times while keeping structural footprint low.	Truncation errors; demands specialized outlier isolation pipelines to handle high-magnitude intermediate activations.
Mishra et al. [45], [29]	2:4 Hardware-Enforced Structural Sparsity Constraints	Sparse Parameter Reduction and Token Pruning Frameworks	Doubles compute throughput by bypassing zero-value parameters using automated index-matching decoding.	Requires strict algorithmic alignment during optimization; fine-grained non-structural sparsity degrades performance.
Shastri et al. [53], [30]	Integrated Co-packaged Silicon Photonics Matrix Engines	Optical Speed Waveguide Matrix Acceleration	Executes massive cross-bar multiplication tasks at the speed of light with minimal thermal dissipation.	High structural overhead from laser source coupling and continuous optoelectronic signal conversions.
Schuman et al. [56], [31]	Asynchronous Event-Driven Neuromorphic Architectures (SNNs)	Continuous Spiking Streams and Local Signal Estimation	Negligible idle-state power draw; excels at event-based spatial audio tracking and localized estimations.	Extremely complex to map standard highly synchronized transformer attention matrices onto asynchronous arrays.
Sheng et al. [53], [32]	Heterogeneous Storage Hierarchies and Token-Paged Memory Caching	High-Throughput Decentralized Edge-AI Generation	Enables execution of large models on consumer-grade computing nodes via structural cache offloading.	Higher execution latency; strictly limited by the localized throughput boundaries of consumer I/O interfaces.
Liu et al. [51]	Array Covariance and Dual-Branch Spatial-Temporal Attention	Multi-Source Spatiotemporal Feature Extraction	Enhanced separation capabilities under multi-source overlapping signals and complex interference.	High structural complexity; scale-up constraints when computing dense covariance matrices over extended frames.
Qian et al. [52]	Sparse Mixture of Experts(SMoE) Parallel Preprocessing	Non-Uniform and Colored Noise Signal Mitigation	High precision under highly unstable, non-ideal noise distributions using localized expert routing.	Increased memory capacity requirements to hold unselected expert weights within the immediate storage array.

4.1 Sub-8-bit Low-Precision Quantization Engines

Quantization is an essential algorithmic technique in order to reduce the von Neumann memory wall by downsizing the numerical accuracy of the weight matrices and network activations [42]. Modern enterprise data centers are no longer based on standard floating-point representation (FP16/BF16) but rather in the sub-8-bit representation space, utilizing INT4 weight-only and mixed FP4 formats [43]. These designs attain as much as 4 times reduction in total memory bandwidth consumption, which allows large underlying models to be fit with small footprints of hardware memory [44].

In order to execute these compressed data types without incurring latency penalties or inflating silicon overhead, current processor architectures include reconfigurable multi-precision Single Instruction Multiple

Data (SIMD) Arithmetic Logic Units (ALUs) [41]. These computing units use hardware driven bit-manipulation and real-time vector scaling factors to perform a variety of low-precision matrix operations using the physical silicon footprint of a single legacy INT16 multiplier block [45], [46].

Nevertheless, the aggressive sub-8-bit quantization results in a serious mathematical truncation error, and loss of precision, which is mostly due to the occurrence of high-magnitude outlier activations in the intermediate transformer layers [42]. In order to maintain model accuracy and convergence, current computing systems use dedicated outlier isolation logic [44]. This hardware layer is able to sample tensor channels continuously, dynamically routing outlying elements through higher precision (FP8 or FP16) parallel execution pathways and letting the rest of the dense matrix blocks continue on the low-precision pipelines [47], [48].

4.2 Hardware-Enforced Structural Sparsity Pipelines

Sparsity approaches take advantage of the statistical redundancy of deep neural networks by detecting and avoiding the zero-value or statistically insignificant values of weight parameters and reducing redundant mathematical operations [45]. Although non-structural (fine-grained) sparsity provides the maximum algorithmic flexibility, it brings about very stochastic, un-predictable memory access patterns, which worsen the performance of parallel systolic execution blocks [45], [46].

In order to match the physical silicon layout with mathematical compression, current hardware platforms impose strict structural sparsity requirements, with the most notable being the 2:4 sparsity configuration [45]. In such spatial pattern, for every sequential block of four weight elements, exactly two values must be zeroed out during the optimization or training phase [46]. Recent processing units integrate this structural constraint with hardened, low-latency index-matching logic that on-the-fly decompresses sparse matrix metadata at the register-file boundary [47], [48].

5. Future Horizons and Open Research Gaps

With silicon-based complementary metal-oxide-semiconductor (CMOS) technology nearing absolute physical and thermodynamic scaling thresholds, the path of microarchitectural design towards generative AI has to change away from incremental optimization to radical paradigm exploration [49].

5.1 Decentralized Edge-AI and Mobile Generative Inference

The excessive carbon footprint, steep cloud bandwidth costs, and the intrinsic vulnerability of data privacy in centralized cloud computing has fueled a structural shift to edge-AI localized inference ecosystems [58]. Direct execution of optimized sub-10-billion parameter generative models on consumer devices requires the development of ultra-low-power computing paradigms in limited thermal envelops 5 Watts [58]. More recent academic and industrial architectures employ dynamic key-value (KV) cache offloading and special token-paged memory allocation schemes to ensure state coherence in execution without overwhelming localized static RAM (SRAM) caches [4], [5].

5.2 Silicon Photonics and Neuromorphic Non-von Neumann Frontiers

In order to avoid the basic capacitive charging penalties of conventional copper digital pipelines, alternative physical substrates are moving off the theoretical model to physical prototypes [53]. Silicon Photonics has also become a high-throughput computing competitor, taking advantage of multi-wavelength light beams over integrated optical waveguides to perform general matrix multiplication (GEMM) computations at the speed of

light [54], [55]. At the same time, Neuromorphic computing architectures, where asynchronous, event-driven compute paths are executed through Spiking Neural Networks (SNNs) are under investigation to replicate the efficiency of the biological brain [56]. Although this has great potential in sustained streaming workloads such as direction-of-arrival (DOA) processing, the mapping of the highly synchronized attention matrices of current transformers to asynchronous neuromorphic logic arrays has yet to be solved algorithmically [56].

5.3 Critical Open Research Gaps

- The 3D-Stacking Thermal Crisis: The intensive combination of 3D-stacked silicon processing die with high-bandwidth memory (HBM) networks forms dense localized heat flux [17]. Designing hardware-level, predictive thermal redistribution schedulers, capable of balancing processing array loads dynamically without inducing execution pipeline stalls is a crucial gap [17].

- Microarchitectural Side-Channel Security: Hardware-level side-channel leakage is very susceptible to specialized deep learning accelerators. Recent studies in the field of security show that by monitoring transient power profiles, electromagnetic emissions, or access latencies of the memory-bus, attackers can recreate proprietary model topologies and secret weight matrices [57].

6. Conclusions

The quick transition regarding generative artificial intelligence and large language models out of cloud-based research structures into omnipresent computational services has revealed structural constraints in the existing semiconductor fabrics. The domain-specific innovations are gradually replacing the traditional general-purpose architectures as shown by the current research [3]. ASIC-based systolic-array ASICs and tensor-processing engines have provided unprecedented matrix- multiplication throughput through hardware-fusing of custom non-linear activation layers [11], [15].

In order to break these structural constraints, reconfigurable gate arrays provide the much-needed runtime agility to dynamically reconfigure memory channels based on changing model parameters and key-value cache footprints [21], [26]. At the same time, non-von Neumann Processing-In-Memory architectures have become an optimal path for memory bound token generation, directly executing calculations in the storage macros to avoid the overheads of data transfer [31], [35].

Moreover, this survey emphasizes that the optimization of special hardware is not possible in isolation, but must be part of continuous, deep commitment to Hardware-Software Co-design paradigm [41]. Native execution of sub-8-bit quantization system, coupled with the strict mathematical structural sparsity pipelines have demonstrated that algorithmic engineering has a direct impact on physical silicon efficiency [42], [45].

In the future, specialized processor design would need to extend beyond the conventional CMOS limits for mitigating systemic thermal catastrophes as well as eliminate microarchitectural side-channel exploits [17], [57]. Finally, future acceleration of artificial intelligence is dependent on co-architecting robust, secure, and thermodynamically viable processing fabrics that can scale with the next generation of neural architectures [1].

Declaration of Conflicting Interests

The authors declare no potential conflicts of interest with respect to the research, authorship and publication of this article.

Funding

The author received no financial support for the research, authorship and publication of this article.

References

1. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," Advances in Neural Information Processing Systems (NeurIPS), vol. 30, pp. 5998–6008, 2017.
2. N. P. Jouppi, C. Young, N. Patil, D. Patterson, and G. Kurian, "A domain-specific architecture for deep neural networks," Communications of the ACM, vol. 61, no. 9, pp. 50–59, 2018.
3. J. L. Hennessy and D. A. Patterson, "A new golden age for computer architecture," Communications of the ACM, vol. 62, no. 2, pp. 48–60, 2019.
4. W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C. Yu, J. Gonzalez, and I. Stoica, "Efficient memory management for large language model serving with pagedattention," Proceedings of the 29th Symposium on Operating Systems Principles (SOSP), pp. 611–626, 2023.
5. Y. Sheng, L. Zheng, B. Yuan, Z. Li, M. Ryabinin, D. Y. Fu, C. Re, and C. Zhang, "High-throughput generative llm inference with flexgen," International Conference on Machine Learning (ICML), pp. 31145–31160, 2023.
6. T. Hoefler, D. Alistarh, T. Ben-Nun, N. Dryden, and A. Jakobs, "Sparsity in deep learning: Pruning and quantization for efficient inference," Journal of Machine Learning Research, vol. 22, no. 1, pp. 112–135, 2021.
7. Elmobark, N., Saad, A., Hasan, S. H., & Badouch, M. (2026). The Hadoop Ecosystem: An Open-Source Framework for Enterprise-Scale Big Data Processing and Analytics. *Engineering Headway*, *35*, 210-226.
8. Saad, A., Hasan, N. F., & Altaher, A. W. (2025). Deep Reinforcement Learning-Based Network Intrusion Prevention in Cloud-Edge Architectures.
9. Elaraby, A., Saad, A., Karamti, H., & Alruwaili, M. (2023). An Optimized Deep Learning Approach for Robust Image Quality Classification. *Traitement du Signal*, *40*(4), 1573.
10. Rasch, M. J., Mackin, C., Le Gallo, M., Chen, A., Fasoli, A., Odermatt, F., ... & Narayanan, V. (2023). Hardware-aware training for large-scale and diverse deep learning inference workloads using in-memory computing-based accelerators. *Nature communications*, *14*(1), 5282.
11. N. P. Jouppi, G. Kurian, S. Li, P. McFarlin, A. Popruzhuk, P. Ranganathan, and D. A. Patterson, "TPU v4: An industrial-scale general-purpose neural network accelerator," Proceedings of the 50th Annual International Symposium on Computer Architecture (ISCA), pp. 1–14, 2023.
12. M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, and G. S. Corrado, "TensorFlow: A system for large-scale machine learning," 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI), pp. 265–283, 2016.
13. S. Yao, J. Cheng, and H. Yang, "Hardware accelerator for deep neural networks: A survey," Journal of Computer Science and Technology, vol. 35, no. 4, pp. 750–772, 2020.
14. Y. S. Shao, J. Clemons, and D. Brooks, "Co-designing accelerators and dataflows for next-generation AI workloads," IEEE Micro, vol. 41, no. 6, pp. 42–52, 2021.
15. H. T. Kung, "Why systolic architectures?" Computer, vol. 15, no. 1, pp. 37–46, 1982.
16. Y. N. Wu, J. S. Emer, and V. Sze, "Accelergy: An architecture-level energy estimation methodology for accelerator designs," IEEE International Conference on Computer-Aided Design (ICCAD), pp. 1–8, 2019.
17. V. Sze, Y. H. Chen, T. J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," Proceedings of the IEEE, vol. 105, no. 12, pp. 2295–2329, 2017.
18. A. Parashar, M. Rhu, and J. Emer, "SCNN: An accelerator for compressed-sparse convolutional neural networks," Proceedings of the 44th Annual International Symposium on Computer Architecture (ISCA), pp. 27–40, 2017.
19. J. Yu, A. Lukefahr, and S. Mahlke, "Scalpel: Customizing DNN execution to hardware layouts via selective layer pruning," Proceedings of the 44th Annual International Symposium on Computer Architecture (ISCA), pp. 613–625, 2017.
20. M. Alwani, H. Chen, and M. Annavaram, "Fused-layer CNN accelerator for embedded systems," Proceedings of the 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), pp. 1–12, 2016.
21. C. Kachris and D. Soudris, "A survey of FPGA-based accelerators for cloud computing," ACM Computing Surveys (CSUR), vol. 49, no. 4, pp. 1–32, 2016.
22. J. Cong, "Customized computing in the post-Moore era," IEEE Design & Test, vol. 35, no. 5, pp. 23–31, 2018.
23. C. Wang, L. Gong, and X. Zhou, "Polyhedral-based data reuse optimization for FPGA-based deep learning accelerators," IEEE Transactions on Computers, vol. 70, no. 12, pp. 2110–2123, 2021.

24. K. Guo, S. Zeng, and H. Yang, "A vector processor for neural network acceleration on FPGA," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 26, no. 12, pp. 2642–2655, 2018.
25. X. Zhang, J. Lu, and D. Chen, "Exploiting residual activation redundancy for memory-efficient FPGA accelerators," IEEE International Symposium on Field-Programmable Custom Computing Machines (FCCM), pp. 105–112, 2020.
26. T. Wang, C. Wang, and G. Sun, "A high-throughput FPGA accelerator for transformers using hierarchical dataflow," IEEE Transactions on Parallel and Distributed Systems, vol. 34, no. 5, pp. 1412–1425, 2023.
27. Y. Guan, H. Liang, and J. Cong, "FPGA-based accelerator for compute-intensive deep learning layers," Proceedings of the 25th International Symposium on Field-Programmable Gate Arrays (FPGA), pp. 85–94, 2017.
28. E. Nurvitadhi, G. Venkatesh, and D. Marr, "Can FPGAs compete with ASICs for AI inference?" Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, pp. 3–12, 2017.
29. L. Gong, C. Wang, and X. Zhou, "MALOC: An automated memory allocation framework for FPGA accelerators," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 39, no. 11, pp. 3815–3826, 2020.
30. S. I. Venieris, A. Kouris, and C. S. Bouganis, "Deploying deep neural networks on FPGAs: A survey and classification," ACM Computing Surveys (CSUR), vol. 51, no. 1, pp. 1–39, 2018.
31. Processing-In-Memory (PIM) Architectures
32. N. Verma, H. Jia, and M. S. W. Chen, "In-memory computing: Advancing from architectural macros to systems-on-chip," IEEE Solid-State Circuits Magazine, vol. 11, no. 4, pp. 43–55, 2019.
33. A. Sebastian, M. Le Gallo, R. Khaddam-Aljameh, and E. Eleftheriou, "Memory-centric computing for AI workloads," Nature Nanotechnology, vol. 15, no. 7, pp. 529–544, 2020.
34. D. Ielmini and H. S. P. Wong, "In-memory computing with resistive memory devices," Nature Electronics, vol. 1, no. 6, pp. 333–343, 2018.
35. S. Li, C. Xu, and Q. Li, "Pinatubo: A processing-in-memory architecture for bulk bitwise operations in non-volatile memory," Proceedings of the 53rd Annual Design Automation Conference (DAC), pp. 1–6, 2016.
36. P. Chi, S. Li, and Y. Xie, "PRIME: A novel processing-in-memory architecture for neural network acceleration in ReRAM-based main memory," Proceedings of the 43rd Annual International Symposium on Computer Architecture (ISCA), pp. 27–39, 2016.
37. L. Chang, Y. Zhang, and H. Yang, "A digital processing-in-memory macro running sub-8-bit operations for edge devices," IEEE Journal of Solid-State Circuits (JSSC), vol. 57, no. 11, pp. 3412–3424, 2022.
38. M. N. Bojnordi and E. Ipek, "Memristive memory processing units for data-intensive computing," Proceedings of the 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), pp. 1–13, 2016.
39. J. Xue, G. Huang, and J. Emer, "A charge-domain mixed-signal computing macro for compressed matrix execution," IEEE Transactions on VLSI Systems, vol. 29, no. 8, pp. 1450–1462, 2021.
40. Q. Guo, X. Guo, and Y. Xie, "Indelible computing: Exploring non-volatile memory macros for continuous learning," IEEE Micro, vol. 42, no. 3, pp. 34–45, 2022.
41. R. Khaddam-Aljameh, M. Le Gallo, and A. Sebastian, "An in-memory computing architecture based on phase-change memory for deep learning acceleration," IEEE Journal on Exploratory Solid-State Computational Devices and Circuits, vol. 7, no. 1, pp. 12–21, 2021.
42. C. De Sa, C. Zhang, K. Olukotun, and C. Ré, "High-throughput machine learning via hardware-software co-design," Communications of the ACM, vol. 63, no. 4, pp. 61–72, 2020.
43. G. Xiao, J. Lin, B. Seznec, H. Wu, J. Yu, and S. Han, "SmoothQuant: Accurate and efficient post-training quantization for large language models," International Conference on Machine Learning (ICML), pp. 38111–38124, 2023.
44. J. Lin, J. Tang, H. Tang, S. Han, et al., "AWQ: Activation-aware weight quantization for llm compression and acceleration," Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 8700–8711, 2024.
45. S. Kim, C. Hooper, A. Gholami, and K. Keutzer, "SqueezeLLM: Dense-sparse quantization for large language models," International Conference on Machine Learning (ICML), pp. 12110–12124, 2023.
46. A. Mishra, J. A. Latorre, and J. M. Alvarez, "Accelerating sparse deep neural networks with 2:4 structural sparsity," Proceedings of the 54th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), pp. 1–13, 2021.
47. J. Pool and F. Yu, "Channel permutations for N:M sparsity," Advances in Neural Information Processing Systems (NeurIPS), vol. 34, pp. 13311–13322, 2021.
48. J. Fang, Y. Ma, and M. Lin, "LLM-Pruner: Structural pruning for large language models," Advances in Neural

Information Processing Systems (NeurIPS), vol. 36, pp. 4410–4425, 2023.
49. B. Liang, J. Han, and L. Huang, "Hardware-driven structural matrix pruning for compressed neural network deployments," IEEE Transactions on Computer-Aided Design, vol. 41, no. 8, pp. 2410–2423, 2022.
50. Z. Wang, G. Geng, and X. Zhou, "Quantized matrix decomposition mechanics for processing edge transformer channels," IEEE Computer Architecture Letters, vol. 22, no. 2, pp. 89–92, 2023.
51. H. You, C. Li, and Y. Lin, "Drawing-in-the-Dark: Pruning convolutional neural networks with structural patterns," IEEE Transactions on Neural Networks and Learning Systems, vol. 33, no. 11, pp. 6512–6524, 2022.
52. T. Liu, G. Zhang, and X. Wang, "CNN structure based on array covariance attention for deep learning classifiers," IEEE Signal Processing Letters, vol. 31, pp. 412–416, 2023.
53. X. Qian, J. Yang, and H. Chen, "Attention based DOA estimation in the presence of unknown nonuniform noise," Signal Processing, vol. 218, Article 109371, 2024.
54. B. J. Shastri, A. N. Tait, and P. R. Prucnal, "Photonics for artificial intelligence and neuromorphic computing," Nature Photonics, vol. 15, no. 2, pp. 102–114, 2021.
55. W. Bogaerts, D. Pérez, and J. Capmany, "Programmable photonic circuits," Nature, vol. 586, no. 7831, pp. 207–216, 2020.
56. H. Zhou, J. Fang, and M. Lin, "Photonic matrix multiplication chips for high-speed neural networks," IEEE Journal of Selected Topics in Quantum Electronics, vol. 26, no. 5, pp. 1–18, 2020.
57. Elmobark, N., Saad, A., Al-Khazalli, A. A. T., & Badouch, M. (2026). Evolving Text Matching: A Systematic Review of Classical and Modern Approaches in the Neural Network. Engineering Headway, 35, 195-209.
58. H. Yu, Y. Wang, and M. Lin, "A survey of security vulnerabilities and mitigation strategies in specialized machine learning hardware cores," IEEE Transactions on Information Forensics and Security, vol. 19, pp. 1432–1448, 2024.
59. J. Zhang, B. Yuan, and C. Zhang, "Edge Intelligence: Paving the last mile of artificial intelligence with edge computing," Proceedings of the IEEE, vol. 107, no. 8, pp. 1738–1762, 2019.
60. K. Shvachko, H. Kuang, and S. Radia, "The Hadoop distributed file system," IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST), pp. 1–10, 2010.
61. A. Shazeer, M. Mirhoseini, K. Zhou, and Q. Le, "Outrageously large neural networks: The sparsely-gated mixture-of-experts layer," International Conference on Learning Representations (ICLR), 2017.