A Multi-Dimensional Evaluation Framework for Matrix-Based AI Accelerators: GPU, FPGA, and ASIC

Kangzhe Peng

2023190502040@std.uestc.edu.cn

Glasgow College, University of Electronic Science and Technology of China, Chengdu, 611731, China

Abstract. The fast developing pace in deep learning is giving pressure on computing hardware continuously. In many practical cases, model size and training cost increase faster than the performance improvement of general-purpose processors. The huge different pace between them makes a gap called "compute gap". Consequently, heterogeneous accelerators such as GPUs, FPGAs, and ASICs have become central to modern AI systems. It is not a easy work to select the right and appropriate accelerator because current studies focus on peak throughput while ignore the latency characteristics, energy efficiency and deployment constraints. This essay introduces a multi-dimensional evaluation framework for matrix-based AI accelerators which concentrated on convolution-dominated workload. Analyzing the represented hardware platform, including NVIDIA A100 GPUs, AMD Xilinx Versal AI Core FPGAs and Google TPU v4 ASICs, and comparing their throughput, determinism, scalability and power consumption. At the same time, in order to provide the guide in selection of hardware in different and detailed scenarios and distinguish compute-bound and memory-bound workloads, the Roofline Model is utilized. The analysis suggests that GPUs and TPUs are generally more suitable for large-scale cloud training, whereas FPGAs tend to be more advantageous in latency-sensitive and energy-constrained edge inference scenarios.

Keywords: AI Hardware Accelerators, Matrix Multiplication (GEMM), Hardware Selection Framework, Compute Gap, Deep Learning

1 Introduction

Deep learning has become a core component of modern artificial intelligence and is widely used in practical systems. Although processor performance continues to improve over time, it is often insufficient to meet the increasing computational demands. The compute gap arises from the growing mismatch between algorithmic requirements and hardware capability. This mismatch is largely unavoidable, as model size and training complexity continue to grow faster than improvements in processor performance.

In the early time, the progress on hardware were usually enough to support new algorithms. This balance did not last for long. As deep learning models became deeper and involved more

ICMEEA 2026, April 24-26, Singapore, Singapore

DOI 10.4108/eai.24-4-2026.2364875

parameters, the required computational cost increased rapidly. One study by Amodei and Hernandez reported that, starting from 2012, the computation needed for state-of-the-art training tasks doubled about every 3.4 months [1]. This rate is far more beyond what Moore's Law could predict. Similar conclusions were also reached by Sevilla et al., who examined the long-term evolution of AI computation and divided it into several phases. Their analysis pointed that there is a transition toward large-scale training, where computational requirements increased by orders of magnitude [2].

More recent industrial cases suggest that this trend has not slowed down. For example, Meta's Llama 3 model, released in 2024, reportedly required on the order of 10^{25} floating-point operations which were used to train its largest configuration [3]. This level of computation is far beyond the computing ability of conventional CPUs or other devices which are not designed for sustained, large-scale and data-parallel workloads. As a result, specialized accelerators such as GPUs, FPGAs and ASICs have become increasingly important in today's AI systems [4].

As illustrated in Fig. 1, there is a widening gap between the computational demands of modern AI models and the performance growth of general-purpose hardware. The red solid line represents the exponential growth in AI model training requirements, which steepened significantly after the introduction of large-scale models like GPT-4 and Llama 3. In contrast, the blue dashed line indicates the hardware performance growth predicted by Moore's Law, which rises linearly and lags significantly behind the algorithmic demand.

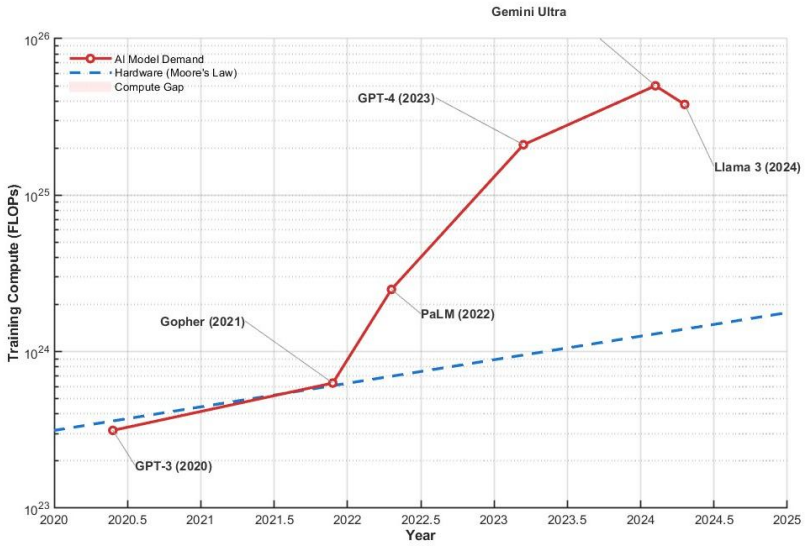

Fig. 1. The widening compute gap between AI model training requirements and hardware performance growth (2012-2024). Data sources: [1], [2], [3].

Although these accelerators technologies and designing methods continue to improve, selecting an appropriate hardware platform is still a challenge. Purkayastha et al. highlighted a critical gap in current research [5]. Many comparative studies only focus on throughput. It cannot provide sufficient guidance for engineers. Requirements always vary so that the consideration

of latency, power consumption and deployment environment is important. This lack of evaluation motivates the framework proposed in this paper, which aims to connect workload characteristics with hardware architecture in a more practical way.

2 Matrix-Based Convolution and Hardware Basics

In this section, it focuses on how convolution operations are transformed into matrix multiplications for efficient computation. In addition, the evaluation metrics used in the later analysis are briefly introduced.

2.1 The Math Behind CNN Acceleration

CNNs are widely used in computer vision because they can capture spatial features effectively. As the depth of a network increases and kernel structures become more complex, the cost of convolution will be high. In a practical implementation, convolution is performed by sliding a kernel across the input feature map, which is not efficient on high-performance hardware.

To address this issue, most accelerators rely on the Im2Col transformation. This method reshapes multi-dimensional input data into two-dimensional matrices. Then, it converts convolution into a general matrix multiplication (GEMM) operation. Once expressed in this form, the computation can take advantage of highly optimized matrix-processing units.

As illustrated in Fig. 2, Im2Col transforms 3D input feature maps into 2D matrices. The left side of the figure demonstrates the sliding window process over the input feature map. The arrows indicate the unrolling process, where the data within each window is flattened into rows on the right side, forming a large matrix (GEMM Input Matrix) that can be processed in parallel by the hardware.

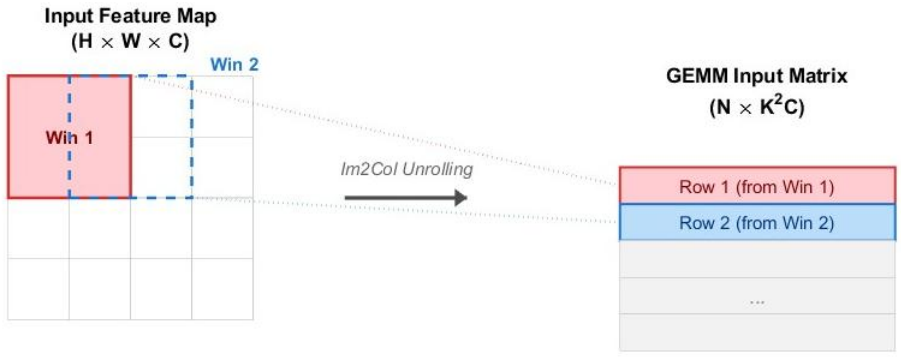

Fig. 2. Schematic diagram of the Im2Col algorithm converting 3D convolution into 2D matrix multiplication.

This observation is important for hardware selection. Since convolution is finally reduced to matrix multiplication, accelerators which are optimized for GEMM tend to achieve higher performance. For example, Jouppi et al. described the Google TPU as being centered around a

large matrix multiply unit composed of 65,536 multiply–accumulate units arranged in a systolic array [6]. More recent designs, such as the flexible diagonal cyclic array proposed by Lee et al., further explore how data movement can be reduced for different kernel shapes [7].

2.2 Key Evaluation Metrics

To compare GPUs, FPGAs and ASICs in a consistent way, four key properties are considered in this study.

Throughput: Throughput is typically measured in TOPS and describes the amount of computation completed within a given time interval. For large workloads with a high degree of parallelism, GPUs generally achieve strong throughput performance.

Energy Efficiency: Energy efficiency is defined as performance per watt. It is a critical factor for both edge devices and large-scale data centers. In practice, ASICs usually deliver the highest energy efficiency, followed by FPGAs, while GPUs tend to exhibit higher power overhead.

Latency: Latency refers to the time required to complete a single task or inference. Comparing the three, FPGAs often provide lower and more predictable latency because the no need of operating system scheduling and instruction-fetch overhead,

Flexibility: Flexibility describes how easily hardware can adapt to new models or algorithmic changes. FPGAs support hardware-level reconfiguration. GPUs rely primarily on software programmability. ASICs offer very limited flexibility once fabricated.

3 In-Depth Architectural Comparison

Compares inner architectures of GPUs, FPGAs and ASICs, and this chapter clarifies why their performance differs on matrix-based workloads.

3.1 GPU Architecture: The A100 Example

The NVIDIA A100 architecture prioritizes a high degree of parallelism. It uses specialized hardware, known as Tensor Cores, to accelerate matrix tasks. These cores operate within a deeply pipelined execution model [8]. They also support mixed-precision matrix-multiply-accumulate instructions. This combination substantially boosts throughput, especially for dense linear algebra workloads. However, the A100 depends on a sophisticated memory hierarchy. At the same time, it requires detailed software-managed scheduling which adds expense. Considering these two factors, small-scale workloads often perform badly. If the data structure is irregular, latency becomes a non-negligible issue.

3.2 FPGA Architecture: The Versal AI Core

FPGAs adopt a unique architecture to meet the demand of huge scale parameters. The Versal AI Core is a case in point. It integrates programmable logic with special AI Engines [9]. This different design gives engineer a chance to build and tailor the custom data paths freely. Furthermore, not like GPUs' data flow, FPGAs allow information deliver directly between processing elements. This special mechanism significantly reduce memory traffic. At the same time, leading to a more deterministic execution. Consequently, for handling real-time and edge

inference tasks, FPGAs are a better choice. In these scenarios, predictable latency is often more critical than raw peak throughput.

3.3 ASIC Architecture: The TPU v4

Discussing ASIC architecture, Google's TPU v4 is a good example. This device is specially projected for machine learning workloads. It utilized systolic arrays to achieve high efficiency. This array can not only enhance data reuse, but also minimizes energy consumption. Furthermore, its interconnect system uses optical circuit switches which allow inter-chip connections to be reconfigured dynamically. This ability is key to scalability and enables the system scale efficiently across large accelerator clusters [10].

3.4 Comprehensive Comparison

As illustrated in Fig. 3, these architectures are compared across five key dimensions. The radar chart visualizes the distinct trade-offs: The red line (ASIC) dominates in Throughput and Energy Efficiency but scores lowest in Flexibility. The green line (GPU) offers a balanced profile with high throughput and flexibility, while the blue line (FPGA) occupies a unique niche, excelling in Latency consistency (implied by Dev Speed/Flexibility in specific contexts) and Flexibility.

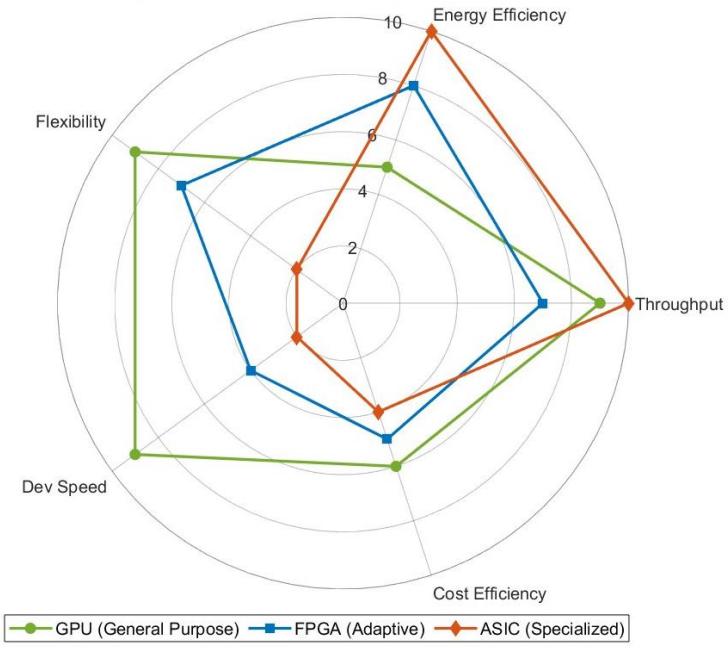

Fig. 3. Multi-dimensional radar chart comparing GPU, FPGA, and ASIC across performance, efficiency, flexibility, development speed, and cost. Data derived from [2], [8], [10].

As the figure shows, GPUs, FPGAs and ASICs show strengths and limitations in different terms. GPUs provide a balance between performance and programmability. ASICs achieve the highest

efficiency for fixed workloads. FPGAs occupy a niche where low latency and configurability are critical.

4 Selection Framework and Scenario Analysis

4.1 The Roofline Evaluation Model

The Roofline Model, introduced by Williams et al., offers an intuitive and efficient way to determine whether a workload is mainly constrained by computational capability or by memory bandwidth [11]. This model can not only link achievable performance to operational intensity, but also identify which hardware features play a more significant role for a specific application.

4.2 Scenario 1: Cloud Training and Large-Scale Inference

In terms of cloud computing environments, throughput and scalability are the key points to assess performance priorities. For Large language models like Llama 3, they tend to show high operational intensity. Since GPUs and TPUs are designed to exploit large-scale parallelism and high memory bandwidth, they can operate efficient execution during training and inference at scale. Therefore, GPUs and TPUs are the two ideal choices for this case.

4.3 Scenario 2: Edge Inference and Real-Time Processing

Edge applications operate in a unique environment. They face two main challenges. First, the power budget is strictly limited. Second, latency requirements are tight. Moving tasks to the cloud is not always a feasible solution. Studies indicate that cloud processing introduces unpredictable delays [12]. Such delays are unacceptable for real-time decision making. In this case, FPGAs and edge ASICs offer specific benefits. They often surpass embedded GPUs. They provide more consistent latency behavior. Furthermore, their energy efficiency is typically superior.

As demonstrated in Fig. 4, there is a distinct trade-off between latency and flexibility across edge hardware architectures. The blue bars represent inference latency, showing that FPGAs (SoC) achieve the lowest latency (under 1ms). Conversely, the orange line tracks energy efficiency, highlighting that Edge ASICs provide the highest efficiency (over 8 TOPS/Watt), significantly outperforming the Embedded GPU.

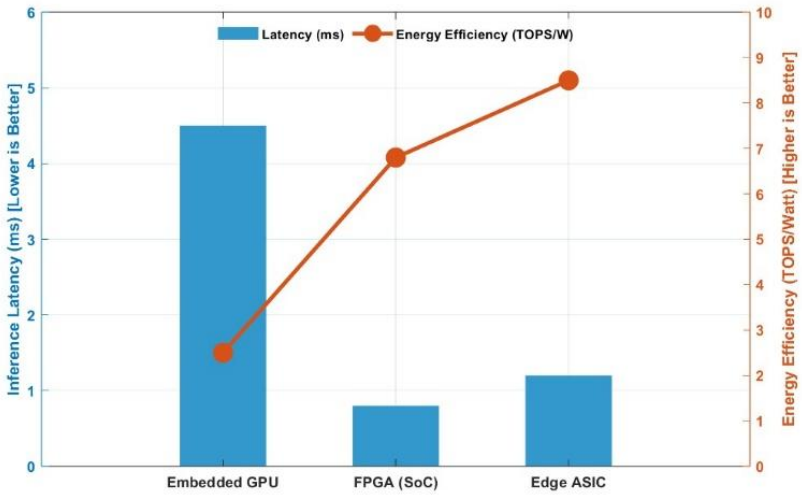

Fig. 4. Comparison of Latency (lower is better) and Energy Efficiency (higher is better) for edge inference workloads across Embedded GPU, FPGA, and ASIC architectures. Data derived from [5], [9], and [12].

5 Future Trends

5.1 Hardware-Aware Pruning

Current neural architectures suffer from severe parameter inefficiency. It is time-wasted because too many parameters will carry millions of redundant weights which are useless to the output. This results in a "computational tax"—wasting cycles on zero-value math. But, this waste is avoidable. As Gale et al. proved, networks can withstand aggressive pruning without collapsing accuracy [13]. This finding need a hardware to change. Simple matrix multipliers are obsolete. To remain viable, next-generation accelerators must implement structured sparsity logic to physically bypass these ineffective operations.

5.2 Processing-in-Memory (PIM)

Data movement is the new bottleneck. Shuttling tensors between external memory and the compute core consumes more energy than the arithmetic itself. Processing-in-Memory (PIM) attempts to reverse this relationship. The goal is to migrate the computation logic directly into the storage arrays. Recent architectural surveys highlight the potential of this approach [14]. It promises massive efficiency gains, particularly for bandwidth-bound workloads. Yet, adoption is slow. Practical deployment faces significant hurdles, primarily due to manufacturing and integration complexities.

6 Conclusion

Deep learning has grown a lot recently. Because of this, hardware is lagging behind. This problem is called the "compute gap". In this paper, a framework is proposed. It is used to test AI accelerators. Many people only look at throughput. It is a mistake to only check speed. In this paper, selecting hardware must also consider latency, energy efficiency and real-world context. If check all these things, the choice becomes clearer. There are differences between chips. For the cloud, GPUs and TPUs are the better choice. They are great for huge scale training jobs. But FPGAs and Edge ASICs are different. They are better for the edge. They work well when power is strictly limited. In the future, it is not a good idea to just make chips bigger. New architecture ideas are needed. Sparsity is one big area. Processing-in-memory is another hope for the future.

References

[1] Amodei, D., Hernandez, D.: AI and compute. OpenAI (2018). https://openai.com/index/ai-and-compute/

[2] Sevilla, J., Heim, L., Hobbhahn, M., Besiroglu, T., Hobbhahn, M., Ho, A.: Compute trends across three eras of machine learning. arXiv preprint arXiv:2202.05924 (2022)

[3] AI@Meta: The Llama 3 herd of models. arXiv:2407.21783 (2024)

[4] Wang, Y.: Artificial-intelligence integrated circuits: Comparison of GPU, FPGA and ASIC. Applied and Computational Engineering. Vol. 4(1), pp. 99–104 (2023)

[5] Purkayastha, A.A., Tharwani, J., Aggarwal, S.: FPGA or GPU? Analyzing comparative research for application-specific guidance. In: IEEE SoutheastCon 2025 (2025)

[6] Jouppi, N.P., Young, C., Patil, N., Patterson, D., Agrawal, G., Bajwa, R., Bates, S., Bhatia, S., Boden, N., Borchers, A., Boyle, R., Cantin, P., Chao, C., Clark, C., Coriell, J., Daley, M., Dau, M., Dean, J., Gelb, B., Yoon, D.H.: In-datacenter performance analysis of a tensor processing unit. In: Proceedings of the 44th Annual International Symposium on Computer Architecture (ISCA). pp. 1–12 (2017)

[7] Lee, D.-Y., Aliev, H., Junaid, M., Park, S.-B., Kim, H.-W., Lee, K.-M., Sim, S.-H.: High-speed CNN accelerator SoC design based on a flexible diagonal cyclic array. Electronics. Vol. 13(8), p. 1564 (2024)

[8] NVIDIA: NVIDIA A100 Tensor Core GPU Architecture. NVIDIA Corporation (2020)

[9] AMD Xilinx: AI Inference with Versal AI Core Series: Solution Brief. Advanced Micro Devices, Inc. (2023)

[10] Jouppi, N.P., Kurian, G., Li, S., Ma, P., Nagarajan, R., Nai, L., Patil, N., Subramanian, S., Swing, A., Towles, B., Young, C., Zhou, X., Zhou, Z., Patterson, D.: TPU v4: An optically reconfigurable supercomputer for machine learning with hardware support for embeddings. In: Proceedings of the 50th Annual International Symposium on Computer Architecture (ISCA). pp. 1–14 (2023)

[11] Williams, S., Waterman, A., Patterson, D.: Roofline: An insightful visual performance model for multicore architectures. Communications of the ACM. Vol. 52(4), pp. 65–76 (2009)

[12] Das, A., Patterson, S., Wittie, M.P.: EdgeBench: Benchmarking edge computing platforms. In: 2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion). pp. 175–180. IEEE (2018)

[13] Gale, T., Elsen, E., Hooker, S.: The state of sparsity in deep neural networks. arXiv preprint arXiv:1902.09574 (2019)
[14] Asifuzzaman, K., Miniskar, N.R., Young, A.R., Liu, F., Vetter, J.S.: A survey on processing-in-memory techniques: Advances and challenges. Memories - Materials, Devices, Circuits and Systems. Vol. 4, p. 100022 (2023)