Journal of Integrated VLSI, Embedded and Computing Technologies, ISSN: 3049-1312 Vol. 2, No. 1, 2025 (pp. 8-19)
RESEARCH ARTICLE ECEJOURNALS.IN

AI Hardware Accelerators: Architectures and Implementation Strategies

K. S. Hyun[1], P. J Min[2], L. H Won[3]*

[1-3]School of Electrical Engineering, Korea Advanced Institute of Science and Technology, Daejeon 34141, South Korea

Keywords:
AI Hardware Accelerators;
Hardware Architecture;
Neural Network Processing;
Parallel Computing;
Implementation Strategies

Corresponding Author Email:
Lehw4on@kaist.ac.kr

DOI: 10.31838/JIVCT/02.01.02

Received : 05.11.2024
Revised : 07.12.2024
Accepted : 05.01.2025

Abstract

Artificial Intelligence also has achieved very rapid advancement, with computational power demands of the kind never before seen. However, continued demand on the specialized hardware has kept right on pace with how complex AI models have gotten. In this article, we dive into AI hardware accelerators architecture, the implementation strategy, and the amazing shift the world of AI got into. That burgeoning field has forced traditional computing architectures to their limit. General purpose processors are generally powerful in many ways other than dealing with the extremely difficult computational demands that AI algorithms require. As a result of this challenge, there is such a new class of hardware: AI accelerators. To enable breakdowns on areas such as computer vision or natural language processing, these are purpose built devices designed to reduce the time and energy required for computing with the AI significantly. To set AI hardware accelerator exploration in the space, we'll evaluate the principles that govern their design, the type of accelerators that currently exist, and how to harness them to their largest deployment. Additionally we will also be envisioning future AI hardware, as well as some emerging trends we believe will define it. If you are a seasoned AI practitioner or just interested to know about the technology of the AI revolution, this detailed guide provides you some insights to the world of AI acceleration.

How to cite this article: Hyun KS, Min PJ, Won LH (2025). AI Hardware Accelerators: Architectures and Implementation Strategies. Journal of Integrated VLSI, Embedded and Computing Technologies, Vol. 2, No. 1, 2025, 8-19

The need for AI hardware acceleration

In spite of the tremendous increase in applications, AI applications, across industries, the old ways of doing computer have failed when it comes to handling AI workloads. While general compute tasks are in the realm of general compute, AI algorithms are extremely general and thus different, and traditional central processor units (CPUs) often fall short when asked to handle this domain. This section explores the relevance and problems that create the requirement and stimuli for specialized AI hardware accelerators .[1-3]

AI Workloads Computational Intensity

AI models have very high computational requirements, and deep learning architectures of AI models particularly so. Because these models have millions or even billions of parameters, it takes hundreds or thousands of mathematical operations to train and to do inference. Specifically, my applications have many matrix multiplications and convolutions, which is the meat and bones of AIs. At large scale, general purpose processors are not capable of performing these computations, and their performance is sluggish and their energy consumption is high.

On top of that, AI workloads are usually inherently parallel, and traditional CPUs are not particularly well suited for taking advantage of that parallelism. It is because AI algorithms lack efficiency in between were unconsumed by computing processes are naturally parallel, and the CPU architecture leaves a lack of number of process for utilization of the computing resources that consequently makes suboptimal

utilization of computing power and very small total performance.[4-6]

Memory Bandwidth Bottleneck

In the same vein, there is a problem of memory bandwidth bottleneck in the AI processing. The demand to have heavy access to huge quantities of data puts heavy stress on the memory system of AI models. However, speed of data transfer between memory and processing units has often been a speed barrier that plagues performance of AI. In particular, this problem is of grave importance when the size of the AI model is significantly larger than the on chip memory, as data transfer between slower off chip memory is necessary quite frequently.

With faster computational capability growing faster than the memory bandwidth, the problem of memory bandwidth is magnified. Although we have seen increases in processing power that have grown rapidly, so too is the memory bandwidth increase has trailed behind the growing gap in the improving efficiency of computing in AI as the relative increase in memory bandwidth has fallen far behind.[7-11]

Energy Efficiency Concerns

Now that energy efficiency is an issue with AI applications everywhere, the desire is to understand MD within the limitations of the energy available on a board. The workloads of AI are power hungry since they run not for the most optimal solution, but for the one that is good enough, resulting in non specific hardware. Both it is a huge source of operational cost as well as the resource constrain to execute AI in energetically constrained environments like edge or mobile devices (Figure 1).

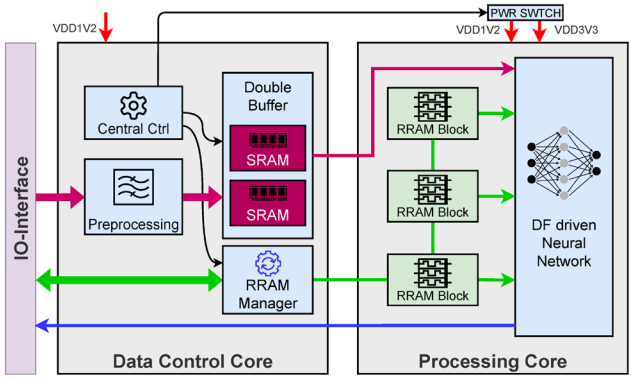

Fig. 1: Real Time Processing Requirements

Since hardware accelerators catering for high performance and low power consumption are being developed, they are designed specifically for the ever demanding requirements of AI processing which needs to provide higher throughput with reduced energy. However, energy effcientiy is crucial to the adoption of AI in a whole range of applications and deployment modes for wide spread deployment.[12-14]

Real Time Processing Requirements

In many of (now) listed AI applications: autonomous driving, robotics, and realtime analytics — low latency processing is a must. Ultimately, these AI models need to make really quick decisions (under a millisecond or two) on the data that comes in (usually). However, the harder the AI model, the more challenging it becomes to work with such stringent latency requirements by using general purpose hardware.

Specifically built hardware accelerators for AI workloads offer dramatic reduction in processing times and thus, near real time or real time performance as required for such line of application. This capability is critical to establishing time critical domain as the playing field where AI can really power up.[15-17]

Scalability and Flexibility

As they get more complicated, AI models need more important scalable and flexible computing solutions in order to improve. First of all, the hardware accelerators have to be capable of quickly serving a range of existing and emerging architectural styles of the AI models, from the edge token execution times to massive language models high throughput in the top of racks of servers (Table 1).

It also ensures that hardware can respond to a new model architecture and computational pattern at the pace of AI innovation in algorithmic space. Flexible accelerator platforms are a hard problem in accelerator design and arise from the need for AI accelerator platforms to be more versatile and programmable.[18-19]

Types of AI Hardware Accelerators

However, unlike the mostly homogeneous world of cloud computing, AI hardware accelerators span a wide range of architectures that have different priorities for different parts of the AI computation. First of all, we will expose some of the major features of major types AI accelerators to provide an overview of what are the strongest points and what are the main applications of the given components.

Table 1: Scalability and Flexibility

Component	Role in AI Hardware	Application in AI
Custom AI Processors	Dedicated processors designed specifically for AI workloads, enhancing computation speed.	Optimizes computation for deep learning, reinforcement learning, and other AI applications.
Parallel Processing Units	Units that execute multiple operations simultaneously, speeding up AI model training and inference.	Enables faster data processing for real-time AI inference in applications like autonomous vehicles.
Memory Hierarchy and Bandwidth	Efficient memory systems and high bandwidth ensure that data is available quickly for processing.	Minimizes memory bottlenecks, improving AI system performance and scalability.
Interconnects and Communication Networks	Ensuring efficient communication between processing units and peripheral devices in AI systems.	Reduces latency and increases throughput, supporting faster AI training cycles.
Energy Efficiency Techniques	Techniques like dynamic voltage scaling and low-power components to reduce energy consumption in AI tasks.	Improves the operational efficiency of AI systems, reducing power consumption while maintaining performance.

Graphics Processing Units (GPUs) is GPUs for these purpose.

Given how GPUs were once created to handle the complex graphics rendering, it is not surprising that GPUs have quickly become engines to speed up AI. They are optimised at paralleling the processing of large data sets, which, more or less, corresponds to how many of the AI algorithms are computationally patterned.[20]

Architecture and Strengths

The thousands of tiny, highly efficient cores in a GPU can work in parallel. Particularly well suited to the matrix operations that dominate most AI workloads, it is. Specifically, modern GPUs include support for specialized tensor cores, which are cores that exist solely for handling the kind of math, in this case deep learning math, but generally. In GPUs, due to the high bandwidth memory and large caches, a reasonable memory usage efficiency in AI processing is allowed by the GPUs memory architecture. The combination of power of parallel processing with power of memory access available in GPUs makes this a very powerful way to perform deep learning, both in training and inference work. Tensor Processing Unit is a customized ASIC (Application Specific Integrated Circuit) on market that targets performing tensor operation in machine learning workload.[21-22]

Architecture and Strengths

Because of their use of the systolic array architecture, TPs apply very well to matrix multiplication and convolution. This design supports high throughput and low latency in the computations performed in an AI. In TPU, the on chip memory reduces huge off chip memory accesses and bandwidth bottlenecks. This means that TPU provides extremely special way of being, that are optimized to work on very common AI operations yet with orders of magnitude better performance as well as orders of magnitude better energy efficiency than typical more general purpose processors.[23]

Use Cases and Limitations

TPUs (on large scale machine learning tasks, especially when scale involves a neural network inference) are especially good for. They use them everywhere on their data centers to run various AI services. Main limitation of TPUs is spesificiy of them. On the upside, they aren't as versatile as GPUs generally and the bonus is incredibly efficient for some types of AI workload. Another uniqueness of the FPGA lies in their full capabilities to acceleration from the field of the AI. These are also devices which have an array of programmable logic block, which can be reconfigured to implement custom digital circuit. Like in the case of FPGAs, they also inherit the reconfigurable nature that allows to design custom accelerator that will be specifically dedicated to the AI algorithms and models targeted. This type of flexibility allows for optimizing AI tasks at collection points, and in principle is very beneficial to performance and energy efficiency. The right AI operations with FPGAs can achieve low latency and high throughput at the expense of performance and in a good fit for real time processing

applications. In addition, they can be keyed to new AI algorithms and model architectures without new hardware.[24]

Use Cases and Limitations

For edge computing scenarios, often FPGAs tend to be high in use on either execution time or energy efficiency. The technology is also useful in rapidly changing fields of AI, in which the capability to change hardware on the fly is an advantage. While programming FPGAs is certainly complex, if FPGAs are to be used, programming skills in hardware description languages need to be in place somehow. In addition, FPGAs are also flexible, however they do not necessarily achieve this raw performance for a given well defined task as well as ASICs do.[25]

Application specific Integrated Circuits (ASICs)

The pinnacle of specialization in AI hardware accelerators pertains to hardware acceleration of AI at the ASIC level. These are chips designed for specific AI tasks or for specific models. As such, the architecture of such an ASIC is entirely customized to AI application. This extreme specialization makes it possible for best performance and energy efficiency. ASICs provide the unique features and optimizations for which it is not possible to be included in general purpose accelerators. ASICs take the elements of a chip not needed for the work they are designed to do and eliminate them or optimize every single aspect of chip design for the specialized work it will be performing (Figure 2).[26]

With more complex and more energy consuming CPUs, ASICs are more and more interesting for precisely defined, stable, on demand AI workloads requiring highest performance and efficiency. They are popular as they are used in high volume products such as those in data centers while having an outstanding energy efficiency such as in mobile devices. ASICs drawback is they are not flexible. After doing an ASIC, it is impossible to reconfigure the ASIC for another task. First, this inflexibility plus high development costs leads ASICs to not be suited to fast evolving AI applications and low volume of deployment.[27]

Neuromorphic Processors

A paradigm shift in AI hardware is neuromorphic processors based on a neural network like structure and function. They are spiking neural networks (SNN) processors, which process information in the manner of the human brain. This type of computation can be significantly energy efficient because the only neurons that 'fire' and computation 'happens' in the case that it does. One specific consideration for neuromorphic architectures which are an important problem is the von Neumann's bottleneck, and in particular, if novel memory technologies that store and process information on the same physical location are added — memristors for example.[28]

Such realtime processing of sensory data is a real need for applications like robotics or autonomous

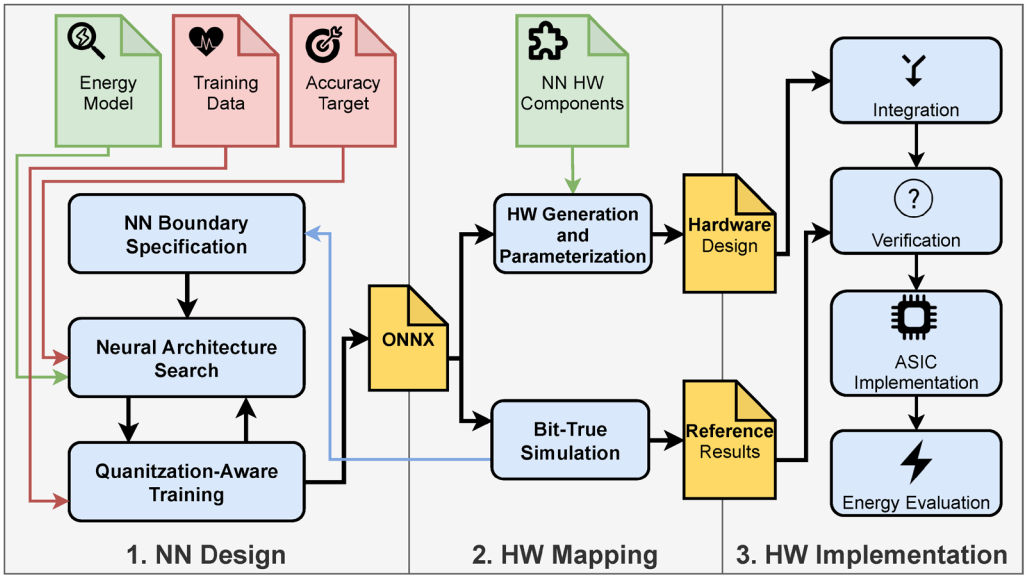

Fig. 2: Application specific Integrated Circuits (ASICs)

vehicles and neuromorphic processors are a good means to address that need. For use in the scenarios of edge computing, they are also evaluated for their energy efficiency. Nevertheless, the field of neuromorphic computing is in its infancy. However, programming models and tools for these processors are less developed than those for traditional computing paradigms and that may thus impede their usage. As AI workloads come in a wide variety of shapes and form, so does landscape of hardware accelerators for AI workload. The hardware must consider the type of acceleration required and therefore requires different types of accelerators, depending on the application of AI such as performance demands, energy limitations, flexibility demands, etc. Great work at AI will lead to advancement of the design of accelerator and combine elements from different approaches to make more powerful and effective AI computing platform.[29]

AI Architectural Principles of Accelerators

Architectural principles leading to AI hardware accelerator designs focus on its performances, energy efficiency and flexibility of AI workloads. We explore these key principles, and how they define the architecture of modern AI accelerators in this section.

Parallelism and Vectorization

The design of AI accelerator is at the core of massive parallelism. Just as with other algorithms of AI, deep learning algorithms have a lot of inherent parallelism: the independent thirds can all be processed in parallel. In principle, they usually have many processing elements or cores running in parallel, thus defining an AI accelerator. This architecture achieves the capability of sequential execution of multiple computations resulting into improved throughput over sequential processing. Here for example, the GPU has a CUDA core which is thousands of parallel floating point operations in a core and the number of the cores in a GPU is so large. Similarly, systolic arrays — a grid of processing elements capable of high parallelized matrix multiplication — are used within TPUs (Table 2).[30]

Many of the AI accelerators use Single Instruction, Multiple Data (SIMD) or Single Instruction, Multiple Thread (SIMT). When the data that needs to be processed is such that a single instruction can be applied to many of them (i.e vector or matrix operations which are commonplace in AI workloads), these approaches are well suited. NVIDIA's GPU architecture is an example of a simt value like they can parallelize AI computations by running multiple threads on the same instruction at different data elements.[31]

Memory Hierachy and Bandwidth Optimization

As the memory intense nature of AI workloads demands that AI accelerators are very efficient at memory access, we derive the cost model for on-chip and off-chip IO data movement using Simplex caching strategy. In the design of accelerators, many strategies are used

Table 2: AI hardware accelerators and their impacts

Strategy	Focus Area	Impact on AI Systems
Optimized Hardware-Software Co-Design	Integrating AI algorithms with hardware design to maximize efficiency and performance.	Improves overall system efficiency by closely coupling software and hardware.
FPGA-Based AI Accelerators	Utilizing FPGAs to create programmable hardware solutions that can be reconfigured for various AI tasks.	Provides flexibility to adapt to different AI tasks, improving the system,Äôs versatility and reconfigurability.
ASIC Development for AI Tasks	Designing Application-Specific Integrated Circuits (ASICs) tailored for specific AI workloads to maximize performance.	Delivers high performance for specific AI applications with low power consumption and reduced latency.
Custom AI-Specific Machine Learning Algorithms	Developing algorithms that are optimized for AI accelerators to improve data processing speeds.	Enables faster training and inference cycles, improving AI model development and deployment.
Integration with Cloud-Based AI Systems	Connecting AI hardware accelerators with cloud systems to enhance computational power and scalability.	Scales AI systems dynamically, ensuring high performance across various computational loads.

to reduce memory bandwidth requirements and reduce the effect of the memory access latency. As often used data lies close to the processing units, large on chip memory or caches typically appear on AI accelerators. This allows to save time collecting, and off chip memory accesses can be much slower and highly energy hungry. For example, Google TPUs prominently include an 'on chip' memory which we call the 'Unified Buffer' for storing intermediate results and parameters, and so ship them less frequently from the off chip memory. Data transfer rate in many accelerators is increased via the use of high bandwidth memory. People use HBM or GDDR memory technologies commonly in GPUs (and other AI accelerators) for the bandwidth they provide to support the AI workload.

Memory Compression, Sparsity Exploitation

Memory compression techniques are also used by some accelerators to optimize memory usage even further, or to use the sparsity in AI models. In practice, the usage of these techniques for data compression and skipping computation on zeros can increase the memory bandwidth and reduce the energy consumed. What is more important is functional units designed for AI operations that are more popular in AI accelerators. The operation currently does complex operation in an efficient way versus the general purpose arithmetic units.

NVIDIA's GPUs have tensor cores for accelerating multiply accumulate operations (typical to many deep learning algorithms). These cores allow mixed precision matrix operations to be much faster than could be run with handling of the mix by traditional floating point units. Matrix multiplications are the core of Google's TPUs capacity, and so they are particularly well suited to systolic arrays. It computes some fraction of the problem and electronics the results to its neighbors; they compute a portion (passing results down); and ultimately the results trickle down the end of the systolic array. Some of the accelerators have hardware units to compute the common activation functions present in neural networks such as a ReLU, a sigmoid and a tanh. Usually, these units are useful as they can perform these operations faster than the general purpose arithmetic unit.

Current focus of design for AI accelerators is dataflow architectures. These architectures try to optimize the flow of data through the processing elements by giving large scale storage, high data movement, and poor computational efficiency. Spatial dataflow architectures are appropriate for many AI algorithms, which regularly have regular computational patterns. This is an expensive bottleneck of AI processing and using this means will save the energy and time spent on data movement. temporal dataflow architectures are based on reusing the data over time, and localizing the data in local memories and moving algorithms over (or just with) data. This approach can (especially) well serve to reduce off chip memory accesses as well as energy efficiency.

Programmability and Flexibility

The operation of an AI accelerator requires some sort of specialization, but there is some degree of flexibility needed in order to support a variety of AI model types and accommodate changing algorithms. Configurable datapaths in other accelerators allow them to configure to run different kinds of operations or precision levels. This gives the accelerator, to switch to different AI models or computational need. The aim of software defined hardware is to strike a balance between the systems with high intensity towards tailoring hardware and those with high flexibility to programming. By configuring positions of the above hardware behavior using software, these designs can be adapted to different AI workloads.

Energy efficiency optimizations

Since AI workloads are so vast, energy efficiency is one of those important considerations in accelerator design. Moreover, many accelerators change their power consumption, dynamically based on dynamic voltage and frequency scaling (DVFS) where their frequency changes as a function of workload demands. This technique can be used by the accelerator to save power when computation intensity is low. To the extent that this is possible, reducing power consumption is to power gate part (or all) of the accelerator with techniques that turn off segments that are not required.[32-33]

Mixed-Precision Computation

Operating at mixed precision — accelerators can use different levels of precision (e.g. 16-bit or 8-bit on operations that need high precision but are not crucially important in producing a result — differs namely from the programmable accelerators which allows all operations being predetermined and deterministic.

On the higher level, we contrast the architectures of GPGPUs and AI accelerators and demonstrate that the architectural decisions which power AI accelerators are a careful balance between specialization to run AI workloads vs. open mindedness that retains flexibility to run arbitrary applications. Resorting to massively parallelism, improving memory access, and exploiting specialized functional units enables such accelerators to offer AI programs performance and efficiency to the level needed for running. As the field of AI expands, these architectural principles will be refined, new ways of thought will be developed, and the march of AI computation will be reversed further.

Hardware-Software Co-design

Hardware – Software Co Design : is one of the most crucial strategies of how to implement AI accelerators. That's about doing hardware and software together to operate correctly with the loads of AI. Specialized compilers rely on translating high level AI models to efficient code for given accelerator architectures. They utilize different techniques of optimization to exhaust what it has to offer in terms of its accelerator resources to use. NVIDIA's CUDA compiler for GPUs not only provides optimizations built for deep learning workloads such as kernel fusion and memory access pattern optimization, but it also offers an embedding strategy that flexibly adapts types to code target (GPU, CPU, etc). Just as with the Google XLA (Accelerated Linear Algebra) compiler, TensorFlow also optimizes on computations onto all types of hardware platform from TPUs to GPUs. One major part in hardware software co design is developing language specific for AI (domain specific languages). These languages help programming the AL algorithm in a way that is easily translatable to the underlying hardware. Nvidia has CUDA, Google has JAX which define a high level way to write accelerator friendly ai code. Balancing the energy and performance of AI Accelerators requires an approach like Quantization. This involves reductions of precision that significantly reduce memory requirement and computational complexity (Figure 3).[34]

Mixed Precision Train and Inference

Support of the many accelerators for mixed precision operations will include operations using different levels of precision in different portions of the model. Mixed precision matrix multiply-accumulate operations that NASA supports are important as they allow training and inference time to be drastically reduced with only a small loss of model accuracy, thanks to NVIDIA;s Tensor Cores. The process is quantized for a licensed model, converting it from high to a lower precision with out retraining as it is trained. this technique can achieve dramatic size and run time reduction on model size and inference latency with essentially no

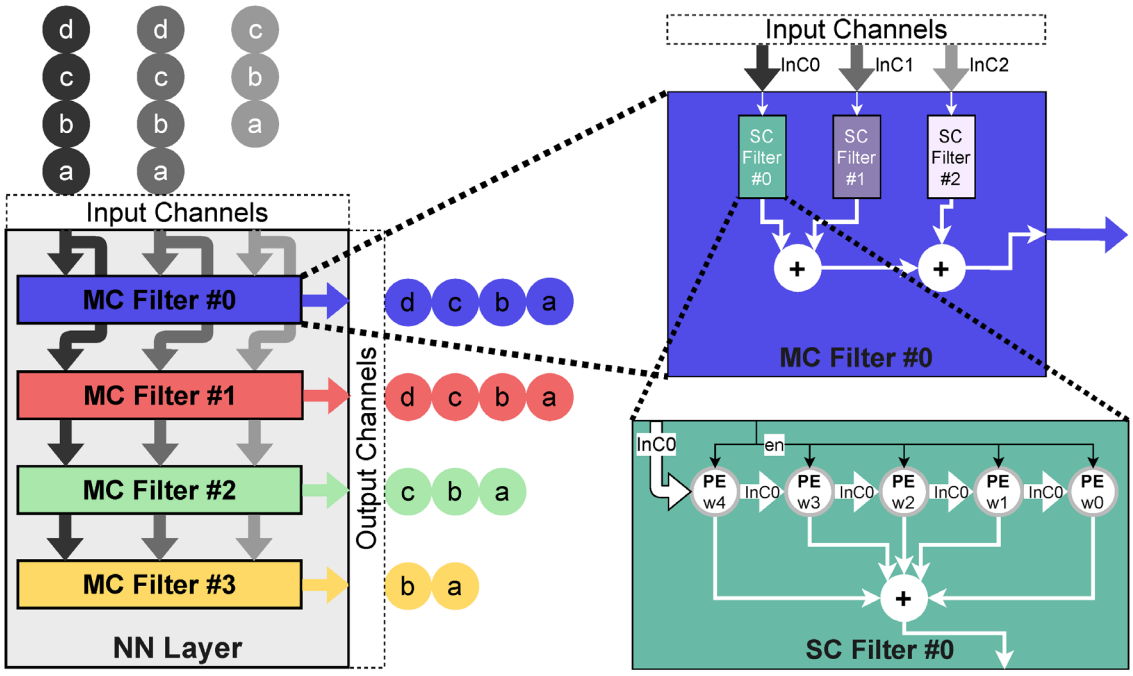

Fig. 3: Mixed Precision Train and Inference

loss in accuracy compared to our baseline. For more aggressive quantization, quantization aware training takes care of quantization effects during its training process. With this I am able to generate models that still begin to have high accuracy, even for precisions like 8 bit or even 4 bit.

But this alleviates the issue of deployment of the AI models to hardware accelerators, by methods like model compression that shrink size and computational intensity of the models. Pruning finds the feat of taking away unnecessary connections or neurons in a neural network. Thus, this can significantly shrink model size and reduce computational requirements without hurting accuracy much. With sparsity, there is an extra complication with accelerator support for sparse tensor operations like Ampere from Nvidia, which can be used as long as the model has been pruned.

Knowledge distillation is a task of training a small, small model which mimics that of a much large, large model. As a result, these techniques produce compact models that can be deployed on resource constrained accelerator. One has to pay attention to different strategies for distributed computing across different accelerators or devices when the scale of the problem is large enough, namely in cases with large scale AI applications.[35-36]

Multi TPUs and Multi GPU Training

The distributed FAI workloads can be supported by frameworks like TensorFlow and PyTorch that provide support for multiple GPUs or TPUs. Thus allowing us to train very large (but maybe not large enough) models on a single device or dramatically cut down training times on smaller models. In federated learning devices / servers are trained with local sample data (also called clients) while the model is trained jointly with respect just to those local data. Specifically, we find that the edge AI application and scenarios with data privacy concern are most appropriate for this approach. It will need to be in order to enable power management and thermal optimization for AI accelerators.[37]

Dynamic Voltage and Frequency Scaling (DVFS)

In DVFS techniques, accelerators have the ability to adapt their clock speeds and voltage levels as required to meet workload requirements without having too much negative effect on performance. Workloads can be scheduled using thermal aware scheduling algorithms that will distribute the workloads to have thermal load balance amongst devices and keep overall performance consistent across devices. In cloud and data center environment, AI accelerator acceleration can be shared with other users or workloads due to virtualization technologies.

In the multi-tenant world, NVIDIA's vGPU offers the opportunity for additional virtual machines to sit on the top of the same physical GPU, making the numbers add up. Such accelerator management plugins for containers orchestration platforms (such as Kubernetes) can effectively deploy and scale AI workloads in a cluster of accelerators. There are special challenges to deployment of AI accelerators at the edge. At the edge, there are also limited computational resources and power available and model quantization, pruning and architecture search are very important for that use case.[38-42]

Hardware Aware Neural Architecture Search (NAS)

NA techniques can be used to specifically take specific characteristics of edge AI accelerators into consideration, and automatically design neural network architectures efficient to deploy on such devices. While benchmarking and continuesearching for the best performance is very important for AI accelerators, especially for it's optimum effciency.

Standardized Benchmarks

Then, you end up with standardized benchmarks like MLPerf, so that that you can compare the performance of a certain type of accelerators for a specific task. These benchmarks are needed to choose the best hardware for these specialty AI workloads. Developers can use vendor provided tools like NVIDIA's Nsight Compute or Intel VTune Profiler to do the analysis on their AI workload performance on any accelerator and pick up optimization opportunities.

There is a complex interaction between hardware design, software optimisation and system level strategies for AI accelerators. With such approaches organizations can optimize their performance, efficiency and flexibility of their AI infrastructure. As the field of AI continues to progress, we will continue to approach these implementation strategies with these types of implementation strategies, which will allow these implementation strategies for more and more powerful and as efficient AI applications in yet another domain.

Performance Evaluation and Benchmarking.

Consequently, one would also need to be able to evaluate performance of AI accelerators in order to pick a hardware and optimize, and therefore compare, alternatives, to fully understand the capabilities of any given AI accelerator solution. In this section, we describe evaluation of AI accelerators based on the methods, metrics and challenges. Typically, performance of AI accelerators is evaluated from a set of important metrics. To really be able to compare and analyze the performance, we need to know these metrics. It is the amount of operations or inferences an accelerator can perform within a time interval. However, it's normally phrased in OPS (operations per second) or IPS (inferences per second). Image per second or token per second, metric commonly used for training workload. For example, metrics for inference in video processing case might be frames per second or queries per second. In the context of this work, we make the definition of latency: the amount of time taken to run a single operation or inference. Hence, in such applications this metric will be very important because real time applications cannot afford slow response times.

In particular, the latency is expressed in terms of milliseconds (ms) and microseconds (μs) as a function of the model complexity and input data size. They measure how much можна computation utilise for per unit of consumed energy. These are the typical metrics of operations per watt (OPS/W) or inferences per watt (IPS/W). For example, these metrics are especially important for edge AI applications and large deployment with high power consumption. The measurement of utilization of accelerator's resources helps to see how well an accelerator is being utilized. In this context that can be metrics like compute utilization (percent of active compute units) and memory bandwidth utilization etc. Utilization is high if it usually means that capabilities of the accelerator have been utilized efficiently, and low utilization could be an indication of scaling problems or problems in implementation acceleration.

Conclusion

By doing so following some low level design decisions taken by Intel's latest server product, we suggest it would have been possible to remove the need for updates to the hyperscalers' specifications. A work load based benchmarks is also another thing we need to benchmark on, however, a good comparison statement is that the standard benchmarks. One of the benefits of KairosDB is that you can run your custom metrics; while many companies build custom benchmarks, which mimic the company's workloads for AI, to troubleshoot. This is the best way to know, how exactly an accelerator would be used in a given use case. It is needed to use the actual world data from the datasets. The funny thing about it is maybe you don't have exactly these kind of easy patterns, these kind of easy edge cases that maybe synthetic data provides. In most of the cases however, you would just want to benchmark the whole AI pipeline from data preprocessing, inference to post processing. End to end evaluation gives you a holistically view of how system performs. And benchmark AI accelerators holds a great potential but with challenges unique enough to require careful thinking to benchmark accurately and fairly.

References

1. Abdelfattah, M. S., Dudziak, Ł., Chau, T., Lee, R., Kim, H., & Lane, N. D. (2020). Best of both worlds: AutoML codesign of a CNN and its hardware accelerator. In *Proceedings of the 57th ACM/IEEE Design Automation Conference (DAC)* (pp. 1–6).
2. Zhang, S., Du, Z., Zhang, L., Lan, H., Liu, S., Li, L., Guo, Q., Chen, T., & Chen, T. (2016). Cambricon-X: An accelerator for sparse neural networks. In *Proceedings of the 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)* (pp. 1–13).
3. Zhou, X., Du, Z., Guo, Q., Liu, S., Liu, C., Wang, C., Zhou, X., Li, L., Chen, T., & Chen, Y. (2018). Cambricon-S: Addressing irregularity in sparse neural networks through a cooperative software/hardware approach. In *Proceedings of the 51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)* (pp. 1–14).
4. Chandramoorthy, N., Swaminathan, K., Cochet, M., Paidimarri, A., Eldridge, S., Joshi, R. V., Ziegler, M. M., Buyuktosunoglu, A., & Bose, P. (2019). Resilient low voltage accelerators for high energy efficiency. In *Proceedings of the 2019 IEEE International Symposium on High-Performance Computer Architecture (HPCA)* (pp. 147–158). https://doi.org/10.1109/HPCA.2019.00034
5. Deng, C., Sun, F., Qian, X., Lin, J., Wang, Z., & Yuan, B. (2019). TIE: Energy-efficient tensor train-based inference engine for deep neural networks. In *Proceedings of the 46th International Symposium on Computer Architecture* (pp. 264–278). https://doi.org/10.1145/3307650.3322251
6. Jang, H., Kim, J., Jo, J. E., Lee, J., & Kim, J. (2019). MnnFast: A fast and scalable system architecture for

memory-augmented neural networks. In *Proceedings of the 46th International Symposium on Computer Architecture* (pp. 250–263). https://doi.org/10.1145/3307650.3322250

7. Rim, D., Kwon, H., & Lee, Y. (2022). Algorithm-hardware co-optimization for cost-efficient ML-based ISP accelerator. In *Proceedings of the 2022 IEEE International Symposium on Circuits and Systems (ISCAS)*. https://doi.org/10.1109/ISCAS48785.2022.9937743
8. Dhilleswararao, P., Boppu, S., Manikandan, M. S., & Cenkeramaddi, L. R. (2022). Efficient hardware architectures for accelerating deep neural networks: Survey. *IEEE Access, 10*, 131788–131828. https://doi.org/10.1109/ACCESS.2022.3229767
9. Sze, V., Chen, Y.-H., Yang, T.-J., & Emer, J. S. (2017). Efficient processing of deep neural networks: A tutorial and survey. *Proceedings of the IEEE, 105*(12), 2295–2329. https://doi.org/10.1109/JPROC.2017.2761740
10. Chen, Y., Luo, T., Liu, S., Zhang, S., He, L., Wang, J., Li, L., Chen, T., Xu, Z., Sun, N., et al. (2014). DaDianNao: A machine-learning supercomputer. In *Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture* (pp. 609–622).
11. Liu, D., Chen, T., Liu, S., Zhou, J., Zhou, S., Teman, O., Feng, X., Zhou, X., & Chen, Y. (2015). PuDianNao: A polyvalent machine learning accelerator. *ACM SIGARCH Computer Architecture News, 43*(3), 369–381.
12. Khadir, M., et al. (2022). QCA-based optimized arithmetic models. In *Proceedings of the 2021 4th International Conference on Recent Trends in Computer Science and Technology (ICRTCST)* (pp. 1–6). https://doi.org/10.1109/ICRTCST52520.2021.9711545
13. Liu, S., Du, Z., Tao, J., Han, D., Luo, T., Xie, Y., Chen, Y., & Chen, T. (2016). Cambricon: An instruction set architecture for neural networks. In *Proceedings of the ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)* (pp. 1–13).
14. Schuman, C. D., Potok, T. E., Patton, R. M., Birdwell, J. D., Dean, M. E., Rose, G. S., & Plank, J. S. (2017). A survey of neuromorphic computing and neural networks in hardware. *CoRR, abs/1705.06963*, 1–4. Retrieved from http://arxiv.org/abs/1705.06963
15. Chen, Y., Xie, Y., Song, L., Chen, F., & Tang, T. (2020). A survey of accelerator architectures for deep neural networks. *Engineering, 6*(3), 264–274. https://doi.org/10.1016/j.eng.2019.12.012
16. Deng, B. L., Li, G., Han, S., Shi, L., & Xie, Y. (2020). Model compression and hardware acceleration for neural networks: A comprehensive survey. *Proceedings of the IEEE, 108*(4), 485–532. https://doi.org/10.1109/JPROC.2020.2978289
17. Nurvitadhi, E., Sim, J., Sheffield, D., Mishra, A., Krishnan, S., & Marr, D. (2016). Accelerating recurrent neural networks in analytics servers: Comparison of FPGA, CPU, GPU, and ASIC. In Proceedings of the 26th International Conference on Field Programmable Logic and Applications (FPL) (pp. 1–4). https://doi.org/10.1109/FPL.2016.7577353
18. Cho, K., Van Merrienboer, B., Bahdanau, D., & Bengio, Y. (2014). On the properties of neural machine translation: Encoder-decoder approaches. arXiv preprint arXiv:1409.1259. Retrieved from https://arxiv.org/abs/1409.1259
19. Tao, J., Thakker, U., Dasika, G., & Beu, J. (2019). Skipping RNN state updates without retraining the original model. In Proceedings of the 1st Workshop on Machine Learning on Edge in Sensor Systems (pp. 31–36).
20. Nurvitadhi, E., Sim, J., Sheffield, D., Mishra, A., Krishnan, S., & Marr, D. (2016). Accelerating recurrent neural networks in analytics servers: Comparison of FPGA, CPU, GPU, and ASIC. In Proceedings of the 26th International Conference on Field Programmable Logic and Applications (FPL) (pp. 1–4). https://doi.org/10.1109/FPL.2016.7577353
21. Cho, K., Van Merrienboer, B., Bahdanau, D., & Bengio, Y. (2014). On the properties of neural machine translation: Encoder-decoder approaches. arXiv preprint arXiv:1409.1259. Retrieved from https://arxiv.org/abs/1409.1259
22. Tao, J., Thakker, U., Dasika, G., & Beu, J. (2019). Skipping RNN state updates without retraining the original model. In Proceedings of the 1st Workshop on Machine Learning on Edge in Sensor Systems (pp. 31–36).
23. Molchanov, P., Tyree, S., Karras, T., Aila, T., & Kautz, J. (2016). Pruning convolutional neural networks for resource-efficient inference. arXiv preprint arXiv:1611.06440. Retrieved from https://arxiv.org/abs/1611.06440
24. Wu, J., Leng, C., Wang, Y., Hu, Q., & Cheng, J. (2016). Quantized convolutional neural networks for mobile devices. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 4820–4828). https://doi.org/10.1109/CVPR.2016.522
25. Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Proceedings of the International Conference on Machine Learning (ICML) (pp. 448–456).
26. Malladhi, N., et al. (2023). Novel architecture of FFT implementation for 5G module using machine learning algorithms. International Journal of System Assurance Engineering and Management, 14(6), 2387–2394. https://doi.org/10.1007/s13198-023-01904-5
27. Wan, D., et al. (2018). TBN: Convolutional neural network with ternary inputs and binary weights. In Proceedings of the European Conference on Computer Vision (ECCV) (pp. 315–332). https://doi.org/10.1007/978-3-030-01234-2_20
28. Rastegari, M., Ordonez, V., Redmon, J., & Farhadi, A. (2016). XNOR-Net: ImageNet classification using binary

convolutional neural networks. In Proceedings of the European Conference on Computer Vision (ECCV) (pp. 525–542). Springer. https://doi.org/10.1007/978-3-319-46493-0_32

29. Jin, J., Liang, C., Wu, T., Zou, L., & Gan, Z. (2021). KDLSQ-BERT: A quantized BERT combining knowledge distillation with learned step size quantization. arXiv preprint arXiv:2101.05938. Retrieved from https://arxiv.org/abs/2101.05938
30. Tambe, T., Zhang, J., Hooper, C., Jia, T., Whatmough, P. N., Zuckerman, J., Santos, M. C. D., Loscalzo, E. J., Giri, D., Shepard, K., Carloni, L., Rush, A., Brooks, D., & Wei, G.-Y. (2023). A 12nm 18.1TFLOPs/W sparse transformer processor with entropy-based early exit, mixed-precision predication, and fine-grained power management. In Proceedings of the 2023 IEEE International Solid-State Circuits Conference (ISSCC) (pp. 342–344). https://doi.org/10.1109/ISSCC42613.2023.10052649
31. Tang, W., Cho, S.-G., Hoang, T. T., Botimer, J., Zhu, W. Q., Chang, C.-C., Lu, C.-H., Zhu, J., Tao, Y., Wei, T., Motwani, N. K., Yalamanchi, M., Yarlagadda, R., Kale, S. R., Flanigan, M., Chan, A., Tran, T., Shumarayev, S., & Zhang, Z. (2024). Arvon: A heterogeneous system-in-package integrating FPGA and DSP chiplets for versatile workload acceleration. IEEE Journal of Solid-State Circuits, 59(4), 1235–1245. https://doi.org/10.1109/JSSC.2024.3301234
32. Song, M., Zhang, J., Chen, H., & Li, T. (2018). Towards efficient microarchitectural design for accelerating unsupervised GAN-based deep learning. In Proceedings of the 2018 IEEE International Symposium on High-Performance Computer Architecture (HPCA) (pp. 66–77). Vienna, Austria. https://doi.org/10.1109/HPCA.2018.00017
33. Yazdanbakhsh, A., Samadi, K., Kim, N. S., & Esmaeilzadeh, H. (2018). GANAX: A unified MIMD-SIMD acceleration for generative adversarial networks. In Proceedings of the 45th Annual International Symposium on Computer Architecture (ISCA) (pp. 650–661). Los Angeles, CA, USA. https://doi.org/10.1109/ISCA.2018.00059
34. Han, S., Kang, J., Mao, H., Hu, Y., Li, X., Li, Y., et al. (2017). ESE: Efficient speech recognition engine with sparse LSTM on FPGA. In Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (pp. 75–84). Monterey, CA, USA. https://doi.org/10.1145/3020078.3021745
35. Devlin, J., Chang, M., Lee, K., & Toutanova, K. (2019). 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 (NAACL-HLT 2019) (Vol. 1, pp. 4171–4186). Minneapolis, MN, USA. https://doi.org/10.18653/v1/N19-1423
36. Naveen, G., et al. (2022). Design of high-performance full adder using 20nm CNTFET technology. In Proceedings of the 2021 4th International Conference on Recent Trends in Computer Science and Technology (ICRTCST). IEEE. https://doi.org/10.1109/ICRTCST52520.2021.9711546
37. Shoeybi, M., Patwary, M., Puri, R., LeGresley, P., Casper, J., & Catanzaro, B. (2019). Megatron-LM: Training multi-billion parameter language models using model parallelism. arXiv preprint arXiv:1909.08053. Retrieved from https://arxiv.org/abs/1909.08053
38. Rodriguez, A., Segal, E., Meiri, E., Fomenko, E., Kim, Y., Shen, H., & Ziv, B. (2018). Lower numerical precision deep learning inference and training. Intel White Paper, 3(1), 1–19.
39. Fang, J., Liu, S., & Zhang, X. (2017). Research on cache partitioning and adaptive replacement policy for CPU-GPU heterogeneous processors. In Proceedings of the 16th International Symposium on Distributed Computing and Applications to Business, Engineering, and Science (DCABES) (pp. 19–22). https://doi.org/10.1109/DCABES.2017.28
40. Lee, J., & Kim, H. (2012). TAP: A TLP-aware cache management policy for a CPU-GPU heterogeneous architecture. In Proceedings of the IEEE International Symposium on High-Performance Computer Architecture (HPCA) (pp. 1–12). https://doi.org/10.1109/HPCA.2012.6168953
41. Bhuiyan, M. A., Pallipuram, V. K., & Smith, M. C. (2010). Acceleration of spiking neural networks in emerging multi-core and GPU architectures. In Proceedings of the IEEE International Symposium on Parallel and Distributed Processing, Workshops and PhD Forum (IPDPSW). https://doi.org/10.1109/IPDPSW.2010.5470915
42. Zhang, X., Gu, N., & Ye, H. (2016). Multi-GPU based recurrent neural networks language model training. In Communications in Computer and Information Science (pp. 484–493). https://doi.org/10.1007/978-3-319-45378-1_44
43. Uvarajan, K. P. (2024). Integration of artificial intelligence in electronics: Enhancing smart devices and systems. *Progress in Electronics and Communication Engineering, 1*(1), 7–12. https://doi.org/10.31838/PECE/01.01.02
44. Uvarajan, K. P. (2024). Advanced modulation schemes for enhancing data throughput in 5G RF communication networks. *SCCTS Journal of Embedded Systems Design and Applications, 1*(1), 7-12. https://doi.org/10.31838/ESA/01.01.02
45. Velliangiri, A. (2024). Security challenges and solutions in IoT-based wireless sensor networks. *Journal of Wireless Sensor Networks and IoT, 1*(1), 8-14. https://doi.org/10.31838/WSNIOT/01.01.02
46. Borhan, M. N. (2025). Exploring smart technologies towards applications across industries. *Innovative Reviews in Engineering and Science, 2*(2), 9-16. https://doi.org/10.31838/INES/02.02.02
47. Sadulla, S. (2024). Techniques and applications for adaptive resource management in reconfigurable computing. *SCCTS Transactions on Reconfigurable Computing, 1*(1), 6-10. https://doi.org/10.31838/RCC/01.01.02
48. Geetha, K. (2024). Advanced fault tolerance mechanisms in embedded systems for automotive safety. *Journal of Integrated VLSI, Embedded and Computing Technologies, 1*(1), 6-10. https://doi.org/10.31838/JIVCT/01.01.02
49. Sathish Kumar, T. M. (2023). Wearable sensors for flexible health monitoring and IoT. *National Journal of RF Engineer-*

ing and Wireless Communication, 1(1), 10-22. https://doi.org/10.31838/RFMW/01.01.02

50. Antoniewicz, B., & Dreyfus, S. (2024). Techniques on controlling bandwidth and energy consumption for 5G and 6G wireless communication systems. *International Journal of Communication and Computer Technologies, 12*(2), 11-20. https://doi.org/10.31838/IJCCTS/12.02.02
51. Tang, U., Krezger, H., & LonnerbyRakob. (2024). Design and validation of 6G antenna for mobile communication. *National Journal of Antennas and Propagation, 6*(1), 6–12.