SoK: Hardware Specialization for AI/ML

Alenkruth Krishnan Murali (jht9sy)[1], Khyati Kiyawat (vyn9mp)[2], and Sabiha Tajdari (jvx2tt)[1]

[1]Ph.D. student, Department of Computer Engineering, University of Virginia
[2]Ph.D. student, Department of Computer Science, University of Virginia

***Abstract*—In this Systematization of Knowledge(SoK) study, we survey the Hardware architectures designed to accelerate Artificial Intelligence(AI) and Machine Learning workloads(ML). We explore two contrasting ML accelerator use cases, data center and the edge in order to learn more about the different design choices based on application. We summarize our learnings and present a brief discussion about the current work in this space and the open problems for the future.**

I. Introduction

Machine Learning has been powering the rapid development of Artificial Intelligence(AI) in the past decade. ML models like Neural Networks(NN) have been successful in learning and estimating high dimension functions. Thereby they are widely used for face recognition, video analysis, object detection and natural language processing. Internally NNs try to mimic the functioning of the human brain. A NN consists of small units called neurons which are connected together to form a network. There are several types of NNs, some of them are Deep Neural Networks(DNN), Convolutional Neural Network(CNN), Recurrent Neural Network(RNN), Long short-term Memory(LSTM), Multi Layer Perceptron(MLP), and Transformers. Figure 1 shows the structure of a DNN.

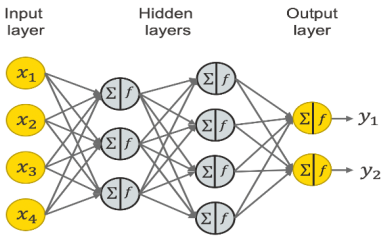

Fig. 1: a DNN schematic

NNs learn and estimate natural functions. For each continuous function, an estimate is obtained using NN. The main problem of NNs is that they require a lot of calculations, especially when the problem is complex or the number of network layers is large. Performing multiple complex calculations (matrix multiplications) in each layer increases the power consumption of the system. Therefore, one of the challenges for the widespread adoption of NNs is to provide a method to reduce power consumption and memory accesses.

Moore's law has been powering the scaling of modern processors wherein for a given die size the speed of processors increases 2X per year. Even though the transistor scaling helps increase memory density and capacity, the memory latency reduces only by 1.1X per year [14]. With time, the gap between processor performance and memory access latency is expected to increase further. The memories are designed to increase the transistors' density, whereas processing units are designed for performance. A widely adopted solution is to introduce caches but they are small and expensive. With the increase in big data applications like NNs which perform rapid memory accesses, the working sets are less likely to fit in the cache. Figure 2 shows the energy spent in data movement.

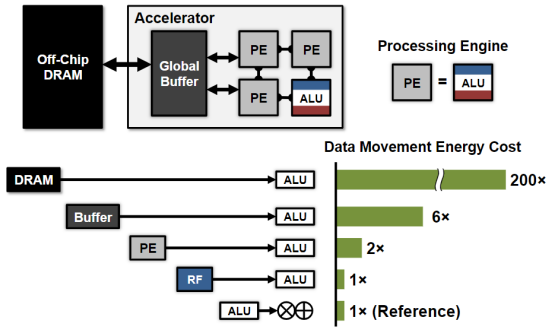

Fig. 2: Energy spent in data movement

The inability of existing general-purpose compute to perform NN computations efficiently necessitated the need for Domain-Specific Accelerators (DSA) [10]. DSAs are tailored to execute the computations present in the trageted domain or application. The DSAs targeted for ML workloads employ Dataflow architectures, exploit data locality and caching techniques to reduce expensive memory accesses, exploit the different parallelism which is exhibited by NNs (as shown in 3) among several other optimizations. These optimizations enable DSAs to be very efficient in accelerating the ML models when compared to contemporary general-purpose CPUs. Nevertheless, the DSAs suffer from a major disadvantage of being designed for a specific domain of applications. When compared with a general-purpose CPU for a diverse workload, the general purpose CPU will perform the best since the DSAs are targeted architectures.

In this study we survey the state of the art ML accelerators that have been designed, or even deployed. While surveying the ML accelerator papers that were published in prominent computer architecture conferences in the past 5-6 years, we realized that we could classify the accelerators into two categories, *1) Accelerators for the edge/mobile computing 2) Accelerators for data center computing*. Since datacenters and

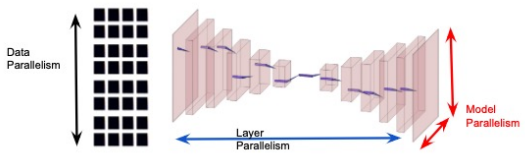

Fig. 3: Parallelism exhibited by Neural Networks

edge devices have contrasting requirements, we felt exploring in these two directions would enable us to uncover interesting design choices and novel hardware/software approaches for Machine Learning workload acceleration. We first describe the accelerators designed for data centers and edge devices. Then we summarize the current work in the ML accelerator space. Finally we list our prediction of where the research community is headed towards and a few open problems that need to be worked upon.

In summary, our study provides

- A taxonomy of the techniques and technologies powering the modern ML accelerators
- Summarizes the key takeaways from the work that has been done so far.
- Provides a the list of open problems that need solutions from the research community.

II. ML Accelerators in the Data Center

The synergy between the large data sets in the cloud and the numerous computers that power it has enabled a renaissance in Machine Learning [21]. As the ML models evolve with a simultaneous increase in their user base, companies have to either aggressively scale out their data centers or accelerate the ML workloads to avoid scaling out. Since scaling out is expensive compared to designing and deploying accelerators in the existing data centers, we have witnessed a plethora of Domain-Specific Accelerators being announced and deployed in the past few years. These accelerators target almost all possible ML models that were listed in I In this section, we survey few of the accelerators that were choses so as to represent each of the different underlying technology platforms. We first state the differences between NL model training and inference. Since data center accelerators have different requirements compared to an edge device, we list the requirements for data center ML inference accelerators. We then classify the data center accelerators based on the underlying technology and elucidate briefly on the novel ideas in their design. We also indicate if the design decisions satisfy or run afoul of the requirements listed earlier. Finally, we summarize our observations.

A. Training and Inference

Training and inference are the 2 major phases in any Machine Learning model. During training, the model is fed with a curated dataset and its hyperparameters are tuned to suit the target application. During inference, we use the trained model to make predictions based on live input data which is subsequently used by the application, usually in making decisions.

The table below lists the major differences between the training and inference of a ML model.

TABLE I: Training vs. Inference

S.No	Training	Inference
1	Frequent memory reads and writes (Forward and back propagation)	Requires only memory reads
2	Distributed training is limited by off-chip bandwidth	Distributed inference is easier to implement
3	High precision operators	Lower precision operators do not affect the accuracy
4	Training is experimental (multiple iterations)	Inference is done only once per input
5	Throughput bound	Latency bound

The weights which are frequently read and updated during the two phases of training, forward propagation, and back-propagation. In inference, we only read the pre-trained weights. Therefore, the architect can minimize the long-latency off-chip memory accesses by storing the weights near the processing unit. Machine learning models use a large number of parameters; the latest Megatron-Turing NLG transformer uses 530Billion parameters [3]. These large models require large on-chip memories and compute power which is infeasible on a single die. Hence, distributed training and inference is practiced. In distributed training/inference, the machine learning model is split into several partitions and then deployed on a single/cluster of accelerators. Inference does not update weights based on other parts of the model, but in training there is a continuous update of weights during the process this makes distributed training to be limited by inter-accelerator communication. Training also makes use of higher precision floating point arithmetic for better accuracy. But once the training is complete, the floating point parameters are converted into integers or floating point with lower precision by *quantization*. Finally, the training operation is an experimental process where the ML engineer tries fine tuning the parameters to best-suit the target application. This necessitates multiple training runs and a higher throughput. Inference is repeated for every request. The overall compute required to train a ML model is less than the compute accumulated over time for inference. This makes inference a suitable target for acceleration when compared to training.

B. Requirements for Inference in the Data Center

① Latency; Service Level Objectives(SLO) and Service Level Agreements(SLA).
② Total Cost of Ownership (TCO) [18].
③ Degree of accelerator specialization.
④ Multi-tenancy, Mixed workload acceleration and Isolation.
⑤ Operand Precision.
⑥ Scalability and Future Proof design.
⑦ Power consumption (TDP) and Cooling [18].
⑧ Programmability and Software Interface.

① A majority of applications which utilize the centers are customer-facing. Therefore, **inference latency is the most important requirement** considered during the design of the accelerators. The architects utilize several techniques like batching the requests [18], [21], [34], exploiting parallelism in an individual request [12], weight pinning where the often accessed weights are stored in on-chip caches reducing long latency memory accesses [7], [18], [21], dataflow architectures [7], [12], [18], [21], [24], [30], [34] and processing-in memory [22].

② The Total Cost of Ownership (TCO) is a factor of the Capital Expense (*CapEx*) and the Operation Expense (*OpEx*). The TCO is amortized over a period of 3-5 years. So, for 3 years the total TCO = CapEx + 3 x OpEx. **Organizations care about the Performance/TCO** rather than just the raw performance. Therefore, architects are required to consider the TCO of the accelerator when designing them.

③ The accelerators designed for acceleration of ML models in the cloud cannot be specific for only one type of ML workload. Designing an accelerator to accelerate only one workload requires individual ML accelerators for every ML model which is infeasible. Hence, **accelerators should be designed to accelerate the common computations present in multiple ML models** rather than accelerating a specific ML model in its entirety.

④ A good accelerator architecture should be able to simultaneously service the requests of multiple users (Multi-tenancy), and simultaneously accelerate the inference of multiple models (Mixed workload). Additionally, the accelerators should provide strong isolation between two users using two different models in the cloud. Every organization has its own proprietary ML models which cannot be exposed to a co-resident user from a different organization. Likewise, it is essential that co-resident users are not able to access each others data. Furthermore, an accelerator which is too specialized will not be able to accelerate inference of multiple models ③. Hence, the architects should design the accelerators in such a way that they can **accommodate multiple tenants and models while providing strong isolation between them**.

⑤ After training and quantization, each ML model uses operands with a precision which helps its accuracy. This requirement forces **accelerators to accommodate operations on a set of different operand types**.

⑥ The DNN workloads evolve with DNN breakthroughs [18]. Since the ML field is rapidly evolving, there are new and improved models being released and adopted quickly. An accelerator which is targeted to one specific computation will become obsolete when a newer ML workload shuns that computation for any reason. Hence, **accelerators should be designed in order to accommodate the evolution of the ML models**.

⑦ Since the inference accelerators in the data center will be serving requests continuously it is necessary that the accelerators have a nominal Thermal Design Power(TDP). While designing the TPUv4i [18], the architects decided to keep the TDP at 175W so that it could be air-cooled in the racks. If accelerators need expensive cooling, it increases the overall OpEx due to the additional power provisioning infrastructure required. **Increased OpEx increases the TCO**.

⑧ It is necessary that the **accelerator should be easy to program**. If the ML engineers are not able to program their ML models in a way to exploit the underlying architecture, the whole exercise of designing accelerators becomes pointless. Software interfaces which enable the ML engineer to work less and translate the ML models from the ML engineer's representation to the accelerator's representation with minimal human effort will gain traction easily.

Training accelerators differ from inference accelerators in the way that they are designed to solve the problems listed in I. We indicate that the accelerator design either satisfies or runs afoul of the requirements listed earlier by including the circled number in the sentences like this ⑩

Note	The requirements listed in this section were either explicitly or implicitly stated in the data center ML accelerator papers which we read [7], [8], [10]–[12], [15], [17]–[24], [28]–[30], [32], [34], [38]. Since, the most requirements are common across the designs we do explicitly cite the paper(s) unless the requirements specified in only that work.

C. Types of Accelerators

The several accelerators deployed in the data centers use different technology platforms to accelerate ML models. One of the most common approach to accelerating ML workloads was to use data-flow architectures which differ from the conventional Von-Neumann architectures which are control-flow in nature. Since the ML accelerators are designed to perform the same computation on a stream of input data with limited control transfer, a data-flow architecture can use the die area saved by not implementing complex control structures can be used for implementing additional computation units. The matrix multiply unit in the Tensor Processing Units [18]–[21], [28], the Neural Function Unit (NFU) in DaDianNao [7], the on-chip AI Accelerator in the IBM Telum CPU [23], the Neural Processing Unit (NPU) in the FPGA accelerator designed by Microsoft Brainwaves team [12], the Mozart Accelerator [34] by SimpleMachines Inc., the Reconfigurable Dataflow Unit(RDU) in SambaNova SN10 [30], and the entire Wafer Scale Engine 2(WSE-2) by Cerebras Systems [24] are examples of Data-flow computing approaches being used at the heart of ML workload accelerators. Additionally, the accelerators implement large on-chip memories to store the weights as well as high bandwidth off-chip memories.

*1) **Application-Specific Integrated Circuit (ASIC)**:* An ASIC is designed and fabricated for one specific application and does not allow you to reprogram or modify it after it is fabricated. Many accelerators that have been deployed in the data centers are ASICs due to their low power consumption and performance.

*a) **DaDianNao - A Machine Learning SuperComputer**:* [7] DaDianNao is one of the earlier works which tried accelerating ML models of 2014. DaDianNao introduces *Weight Pinning*, a technique wherein the frequently accessed parameters; weights, are stored in the on-chip memory ①. Weight pinning reduces the need of accessing off-chip memory for the weights during every epoch. The DaDianNao chip is split into 16 tiles and each tile had a Neural Function Unit (NFU), a data-flow processor tuned for Multiply and Accumulate(MAC) ③ and its private 2MB on-chip memory. The 16 tiles also had a common central on-chip memory of 4MB and uses 16W of power ⑦. On-chip networks were implemented for faster access of these memories. Additionally, the on-chip memory was implemented as embedded DRAM and not as SRAM owing to it better density. DaDianNao being an early academic work suffered from multiple shortcomings. The compute resources on the chip is low and would be insufficent for the current large ML models ⑥. DadianNao also did not have a software infrastructure built for the accelerator ⑧. Nevertheless, DaDianNao had a significant impact on the design of the accelerators that were designed in the years after 2014.

*b) **Tensor Processing Unit(TPU) - Google**:* The TPUs are multiple generations of ML training and Inference Accelerators designed and deployed by Google in their data centers. TPU v1 was designed to accelerate inference. Later they designed two generations of training accelerators and then designed the v4i inference accelerator.

*c) **TPUv1 and TPUv4i**:* [18], [21] Over the two generations of inference TPUs, the microarchitecture and architecture of the accelerators have changed drastically. In the latest inference accelerator, there is 128MB on-chip memory which is used for weight reuse. The first generation of TPU had a 28MiB on-chip memory ①. The v4i also has four 128x128 matrix multipliers for the batched inference while the v1 had one 256x256 multiplier ③ ⑥. The v4i supports Google **B**rain**F**loat16 and FP16 operands ⑤ and was designed to accelerate the newer transformer ML models and RNN models with heavy emphasis on the on-chip interconnect, and inter-TPU interconnects ⑥. v4i houses Tensor Direct Memory Access(DMA) Engines which help hide the off-chip memory access latencies ① ⑥. v4i has HBM memory when compared to the DDR3 DRAM in v1 ① ⑥. The XLA compiler compiles the ML workloads such that they can exploit the processing capabilities of the TPU ⑧. TPU also makes use of a 322b VLIW CISC ISA since longer instructions can reduce the number of instruction memory accesses ⑧. The large on-chip memory on the v4i allows multi-tenancy and multiple workload acceleration with software providing isolation ④. The TPU v1 consumed a nominal 75W of power and v4i consumes 175W making it a suitable candidate for air cooling.

*d) **TPUv2 and TPUv3** [28]:* The training TPUs, v2 and v3 were built based on the architecture of v1 inference accelerator. The major changes included changing the activation pipeline into a vector compute unit to support multiple vector computations, a larger on-chip vector scratchpad memory and HBM main memory, a larger Matrix Multiplication Unit and the introduction of on and off-chip networking for distributed training. TPU v3 also introduced two TPUs in one accelerator die enabling easier distributed training of larger models at lower off-chip data transfer latencies. The TPUs also support distributed training of models across the data center network. The training chips make use of the same XLA Compiler and VLIW ISA.

*e) **On-Chip AI Accelerator - IBM Telum** [23]:* IBM introduced an on-chip AI accelerator for the Telum server CPU. The accelerator interfaces with the fast shared L2 cache and all cores have access to the AI inference accelerator ①. The accelerator does not use instruction chaining and server inference requests one at a time. The on-chip accelerator was chosen for privacy and data security ④. The inference accelerator is generic, uses DLFLT16(Deep Learning Float) operands, and accelerates the MAC operations in the ML models with a systolic array multiplier ③ ⑥ ⑤. The ISA of the CPU was appended with ML specific instructions, and the firmware controls the offloading of instructions ⑧. Since, the accelerator is on-chip there are no additional costs in the design and deployment ②. Moreover, the accelerator does not influence the power and clock frequency of the base cores ⑦.

*f) **Tesla Dojo - ExaScale Training Supercomputer** [38]:* Tesla introduced a CPU with AI compute capabilities. Instead if using an on-chip or off-chip accelerator, the CPU uses an ISA which is designed for ML applications ⑧ ③. The CPU has wide Multi-threaded Vector units and Scalar units, and a large SRAM ① ②. The core has an on-chip Network router which helps efficient scaling to large number of CPU clusters for distributed training ⑥. Similar to the case of IBM Telum, there are no additional design and deployment costs since the acceleration is part of the CPU and the software manages the ML acceleration ② ⑦.

*2) **Field Programmable Gate Array - FPGA**:* FPGAs are flexible and configurable compute substrates which can be configured to perform the required tasks. FPGAs are mostly utilized for prototyping designs during the ASIC design cycle. But, recently FPGAs have seen traction due to their ability to be configured based on the workload required. This is particularly interesting for ML acceleration since a one-time investment in FPGAs which can be reconfigured to accelerate any ML workload is cheaper when compared to an ASIC which might become obsolete and is expensive to fabricate ② ③ ⑥. Modern FPGAs are power efficient and have decently good performance ⑦. Nevertheless, ASICs will always dominate the performance/W metrics due to their application specific nature.

*a) **Project Brainwave - Microsoft** [12]:* The Accelerator uses FPGAs to implement a Neural Processing Unit(NPU), a dataflow microarchitecture which is configurable during compilation (parameters like operand precision can be configured) ⑤. The NPU uses 125W of power ⑦ exploits parallelism per request unlike the other accelerators which batch the requests. Batching requests increases per request latency since the scheduler should wait for sufficient requests before issuing.

Exploiting per request parallelism is particularly important to minimize latency ①. The accelerator uses a single threaded SIMD ISA and the software interface takes care of mapping the compute onto the Matrix-Vector compute engines ⑧. Finally, the accelerator can be scaled to include as many NPUs the model would need.

b) ***Mipsology with Xilinx FPGAs*** *[17]:* Mipsology is software-based startup which provides software suites to map the ML models onto Xilinx FPGAs. The common computations are accelerated and the programmer does not need to know about the FPGA to implement their model on it. The accelerators can be scaled to *N* FPGA accelerators cards present in the server or the data center.

3) ***Coarse Grain Reconfigurable Array - CGRA****:* Similar to FPGAs, CGRAs have compute units like tiny CPUs embedded in a network of interconnects which can be configured to suit the application. CGRAs have the similar advantages in ML acceleration like FPGAs ② ③ ⑥ ⑦, and they enable designing dataflow architectures by reconfiguring the on-chip network to route the operands to the execution units in a streamlined fashion. The accelerators discussed in the section are similar to CGRAs in the aspect that they are made of a network of elementary compute units. But, they differ from CGRAs by the fact that they have tiny memory units (to alleviate latency) on the chip in addition to the compute units. The on-chip network is configured using the software to route the inference/training data during operation. The software also takes care of the memory allocation and scheduling of the requests so that the user does not have to know about the underlying architecture.

a) ***Mozart - Reuse Exposed Dataflow*** *[34]:* Mozart uses a tile based architecture of Configurable Circuit Switch Compute Arrays (CSCA) each containing 8 Functional Units. The functional units within the CSCA can be configured to accelerate common computations present in ML workloads ③ using the software ⑧ similar to the way a CGRA is configured. Mozat exploits parallelism using batches of size 4 ①. Mozart also tries to reuse the data fetched from main-memory to a maximum extent (Re-use Exposed Dataflow) and uses vector gather/scatter instructions for efficient memory accesses.

b) ***SambaNova SN10 - Reconfigurable Dataflow*** *[30]:* The SambaNova SN10 has compute and memory units spread out interfaced with a programmable interconnect. The programmable interconnect in the dataflow architecture helps limit the latency of inference since the compute units and their interconnections can be chosen for shortest latency ①. The dataflow architectures implemented by the software during compilation can be used to accelerate any workload (Graph processing, SQL queries, Genomics) ③ ⑧. The weights are stored in the memory unit and the compute units multi-stage pipelines with SIMD capability. The power of the accelerator lies in the software which orchestrates the entire dataflow. In addition to providing opportunities to accelerate multiple workloads, the SambaNova SN10 can accomodate multiple applications and users with strong isolation with the help of the software ④.

4) ***Processing in Memory****:* Processing in Memory (PIM) has emerged among the solutions to alleviate the constraints imposed by the memory wall [36]. Since ML training and inference require a lot of memory access, PIM is an attractive solution for ML acceleration.

a) ***Samsung AquaBolt-XL****:* The Samsung Aquabolt-XL is a HBM2 memory array with a programmable processing unit embedded at the I/O boundary of a bank The processing unit can access multiple banks in parallel and the the software stack controls and schedules the operations performed on the data fetched from the banks. The processing units contain SIMD floating point addition and multiplication units which can be used to perform MAC operations which is common in ML workloads. Even though the processing units are not powerful to perform computations at high throughput, tasks like data pre/post processing can be offloaded to them.

b) ***Memristors****:* Memristor is a circuit capable of performing matrix multiplication calculations at high speed [9].

The Figure 4 shows the model and the V-I curve of the crossbar Memristor module.

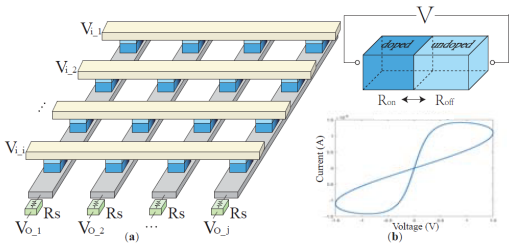

Fig. 4: Memristor

By setting the voltage to the value of N_i and having the matrix W, the product of the matrix can be obtained with the help of the resistance R_i and reading the current. In addition to the fact that the entire computation is performed in memory (no expensive memory access), Matrix multiplication is parallelized.

5) ***Graphic Processing Units (GPU)****:* GPUs have been dominating the ML acceleration space due their inherent parallel nature. Initially GPUs were designed for high throughput. But with the low latency requirement for inference applications, there have been multiple optimizations to reduce the latency of GPUs making it a prime candidate for inference acceleration.

a) ***NVIDIA Hopper GPU****:* GPUs have multiple streaming multiprocessors which perform parallel matrix multiplications. The requests are batched for high utilization and better latency ①. GPUs scale well since they accelerate the matrix multiplication operations which are at the heart of all ML workloads ③ ⑥. The software batches the requests and schedules the execution⑧ in addition to hosting multiple users and providing isolation ④. Moreover, the Hopper GPU architecture also has a Tensor DMA engine, transformer engine, custom dynamic programming instructions, support for HBM3 memory, high speed Multi-GPU I/O, and high speed

networking capabilities. These features help accelerate ML workloads. NVIDIA also enables distributed training of large models like transformers by clustering of GPUs present in the entire data center ⑥.

*6) **Wafer-Scale Compute**:*

*a) **Cerebras - Wafer Scale Engine 2**:* The Cerebras Wafer Scale Engine - 2 (WSE2) is a single chip fabricated on an entire silicon wafer. This allows easier scaling for larger models and also alleviates the need for fast communication between accelerator nodes which uses up large die area in the other single-die accelerators ⑥. The individual compute units called Sparse Linear Algebra Compute (SLAC) cores are connected with a high bandwidth on-chip programmable interconnect fabric. The compute cores have on-chip SRAMs for low latency and high throughput, are optimized for linear algebra, and do not operate on zero values (sparsity) unlike most of the other accelerators ③. Using software the fabric interconnect can be programmed to provide an optimal dataflow path for every ML model①, ⑧. The major limitation of the WSE2 is that it requires a separate infrastructure for powering and cooling it ⑦ ② unlike the other accelerators which use PCIe slots present in conventional servers.

D. Observations

We list our observations from studying the architectural choices and the requirements influencing them in the previously listed accelerators.

- **Accelerators are a function of hardware and software**. One cannot exist without the other.
- **Exploiting parallelism** is the cornerstone to efficient acceleration.
- **Startups tend to design accelerators targeting a large ML application space** while **established organizations accelerate only the workloads their products/customers need.**
- **Software abstracts the underlying architecture** to the programmer enabling rapid programmability.
- **Dataflow architectures** dominate the ML accelerator space. **Reconfigurable technologies like FPGAs and CGRAs are gaining traction due to their configurability**.
- On-chip memory capacity increase is key to better latency. **Memory access latency is the Achilles's heel to acceleration**.
- **On-chip and off-chip interconnects play a major role in allowing accelerators to scale**.
- **Accelerators use custom data types** for efficient acceleration and for lower power consumption.

III. ML Accelerators in the Edge

We are witnessing a big bloom in smart devices in recent years and it is expected to increase. Most of these smart devices and embedded systems are integrated with sensors and perform some kind of feature extraction. Due to the large amounts of high-dimension data, different machine learning techniques and AI methods are applied to extract and learn hidden features. Applications like face recognition in mobile phones, object detection in autonomous navigation, and video analysis in surveillance heavily rely on CNNs and DNNs. Speech recognition and natural language processing are other branches of AI deployed in edge devices like mobile phone applications and voice assistants etc.

Some of the above mentioned applications require real-time responses and hence are latency-critical in nature. Therefore, computations are brought closer to the edge to reduce the latency of operations. However, edge devices face another issue of operating in a power-constrained environment as most of them are battery-operated. Unlike data centers where thermal cooling drives the research for low-power accelerators, edge devices require accelerators that utilize low-power techniques to lower the dynamic power consumption.

As established earlier in Section I, data movement is the most power-expensive step especially when the data is stored far from the compute logic [37]. We also saw that the most frequently used operation in AI/ML is MAC (Multiply and Accumulate). The limited power budget and the nature of the most-common operation i.e., MAC, open up many directions to rethink the way data is stored and used. First, to reduce data movement, keep the more frequently used data near the compute logic. Second, reuse the data already available in the logic to reduce memory accesses. Third, reduce the amount of computation itself.

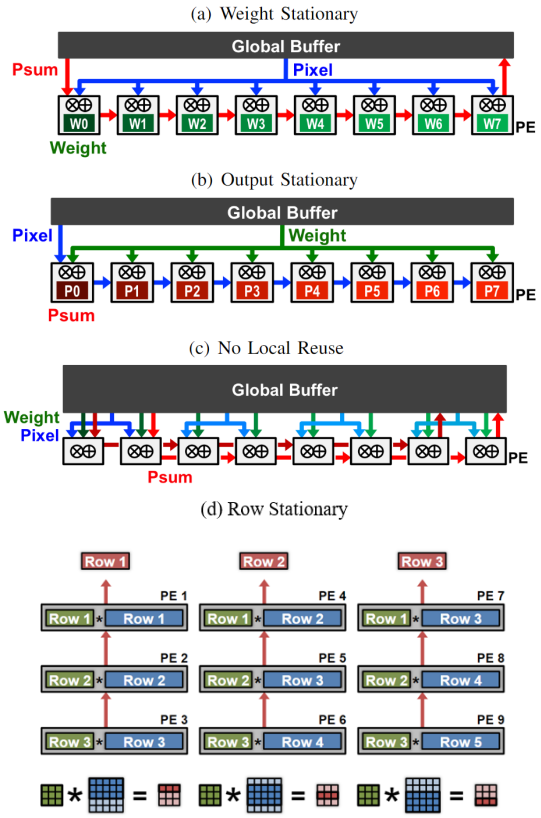

Fig. 5: Dataflows in NN acceleraotrs

Researchers have come up with different dataflows to leverage maximum data reuse as shown in 5 and listed below:

- **Weight Stationary (WS)**: Model weights once learnt are fixed and can be stored in the register file of the processing element. All the inputs and partial sums can move through the global buffer.
- **Output Stationary (OS)**: Another approach is to keep the output in the register file, which can then accumulate the partial sums and avoid their movement
- **No Local Reuse (NLR)**: With strict space constraints, small registers can't hold local values. Therefore, all the local space can be used as the global buffer.
- **Row Stationary (RS)**: Proposed in [5], maximizes input data reuse (i.e. filters and feature maps) and at the same time minimizes partial sum accumulation cost. These minimize the total energy consumption.

Eyeriss [5] shows that when compared with other dataflows such as OS, WS, and NLR using AlexNet as a benchmark, the RS dataflow is 1.4× to 2.5× more energy efficient in convolutional layers, and at least 1.3× more energy efficient in fully-connected layers for batch sizes of at least 16. As we see, just **changing the dataflows impacts the overall energy consumption** of the accelerators. This implies that the dataflows can further be optimized based on the CNN/DNN layers and characteristics of input data, filter, and feature maps. Furthermore, the PEs with larger sizes will always help reduce data movements and frequent memory accesses, given you have enough area on the chip.

The other important aspect when improving the performance of accelerators at the edge is accuracy. When we want to improve accuracy through the use of DNNs, we will still have problems with the lack of resources if we use this on-device method [4]. It is possible to come up with a solution to this problem by using lightweight or simpler models [16], [25]. Since edge nodes are not as powerful as cloud nodes, we need to minimize the number of computations and keep the model size small by using as few trainable parameters as possible [27].

To meet the energy requirements of battery-operated devices, the designers can use techniques such as data quantization and pruning without compromising much on accuracy. Minerva [31] proposes **an automated co-design approach across the algorithm, architecture, and circuit levels** to design and deploy efficient DNN hardware accelerators in power-constrained environments. It consists of five stages:

- Stage 1 explores and finalizes the baseline machine learning algorithm;
- Stage 2 explores microarchitectural optimizations using Aladdin [35] to fix the memory bandwidth, loop level parallelism, clock frequency, etc;
- Stage 3 analyzes data type precision requirements and performs data quantization;
- Stage 4 analyzes pruning statistics of the algorithm and applies operation pruning;
- Stage 5 introduces domain-aware fault mitigation techniques that facilitate SRAM supply voltage reduction.

Expanding further on Stage 4, an important observation here is the data and AI models are sparse. There are certain weights in the neural network layers that are just zero or insignificant. **Getting rid of these insignificant weights** not only reduces storage issues but also reduces unnecessary data movements thereby saving power. [40] also came up with pruning methods for each layer that is guided by the energy consumption of a CNN.

We have talked about two very important ways to design an AI accelerator for the edge. One focused on maximum data reuse thereby reducing power, and the other tried to apply optimization methods such as quantization and pruning to reduce the overall power consumption. All of these techniques cannot be fixed for every model, CNNs and DNNs come with varied model sizes, and the amount and type of data they operate on are diverse. We need **solutions that are more flexible** with input data, NN models, and sparsity. Eyeriss v2 [6] is one such DNN accelerator architecture that has a highly flexible on-chip network called "hierarchical mesh" to address different data types, shapes, sizes, and sparsity of different layers. It adapts to the different amounts of data reuse and bandwidth requirements to better utilize computing resources.

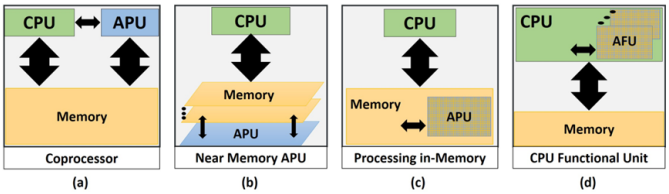

Fig. 6: Where to accelerate?

Accelerating data-intensive and parallel workloads definitely helps overcome the memory wall, reduces overall energy consumption, and lowers the latency of operations. However, **where to accelerate** is still a question of concern; be it in the memory, near memory, on the chip as a co-processor, or as a functional unit in the processor core as shown in 6. Similar to Dadiannao [7], [39] also advocates having functional units in the processor core termed as "Processing with memory". The only difference is that the former used the idea to accelerate NN models in data-center servers whereas the latter proposed the idea for edge processors. The key idea here is to have smaller models (100K parameters as compared to around billions of parameters in data-center servers) that can fit in on-chip AFUs (AI functional units). There are communities like tinyML that push towards reducing the ML model sizes to cater to the resource constraints of embedded processors.

Moreover, edge computing does not only indicates computing on a small device, but it can also have computing shared anywhere among cloud nodes, edge servers, and edge nodes. Concepts like fog computing, mist computing, and in-network computing are also becoming popular. Federated learning is another edge computing method that allows models to be

trained on client-send and share the final models and gradients with the server. The server then generalizes the model and applies the training from all the clients to improve the quality of results for each client. This is an effective approach to **protect clients' privacy** from various adversarial attacks. However, the management of existing devices in terms of computing between the end nodes and the server is still a challenge that needs to be dealt with efficiently, depending on the application at hand.

To summarize, the key metrics for building edge/embedded AI/ML accelerators are:

① Reduced total power consumption i.e. high TOPS/W (Tera Operations per Watt)
② Good quality of results with the required accuracy
③ High throughput for applications that deal with high-volume data analytics
④ Low latency for real-time applications
⑤ Low hardware cost for designing and manufacturing
⑥ High flexibility due to evolving AI/ML models and workloads
⑦ Should address privacy concerns because most edge devices operate on sensitive personal data

Many startups are emerging like SIMa.ai, Esperanto Technologies, Hailo.ai, etc to design hardware specialized for edge AI. Several other edge AI processors like GAP8 of Greenwave Technologies [1] have emerged from the open-source PULP (Parallel Ultra Low Power) platform of RISC-V. [2] shows a joint effort between RISC-V and tinyML communities to develop a configurable edge AI accelerator that leverages the best aspects of RISC-V and eFPGA (embedded FPGA) in an open-source hardware design setup. Nonetheless, coming up with a new accelerator for a specific application and workload is always a challenge which we discuss in the next section.

IV. Design Space Exploration

Developing any hardware requires a large number of man-hours and there is a hefty cost associated with designing, verifying, manufacturing, and testing any hardware. This implies one should not fix hardware for just one application or AI model. Since, the workloads, applications, and even AI models are continuously evolving. The hardware accelerators should be scalable, flexible, and programmable. However, this poses an issue for optimization, designing solutions for the required computation is much more efficient than designing a generalized architecture. Another aspect discussed in the previous sections is where should the acceleration happen, i.e. near the memory or near the processor. There are no standard metrics to compare to accelerators because there are so different in terms of the algorithm used, target applications, and computing resource constraints. We observe the following an accelerator architect can face:

- Generalization vs Specialization
- Bring data near compute unit vs Bring compute near data
- Specialized components vs Specialized instructions
- Online learning vs Offline learning

An exhaustive design space exploration can be a solution to this dilemma. However, researchers need more standardized and open-source frameworks to compare and integrate their accelerators with the existing hardware. We do see efforts in this direction; ESP [26] is an open source platform to integrate heterogenous SoCs, this has different abstraction levels and facilitates ease of usage for ML engineers as well; ScaleSim [33] is another simulator specially developed for CNN accelerators. There are many other simulators designed, however, most of them are developed in-house and therefore have low fidelity and low adoption by the community. There is no one ring that fits all, we need to drive combined efforts from software and hardware community to bridge the gap and develop standardized simulators framework and platforms.

V. Past and Present

As of date there has been several hundred accelerators being built and deployed for acceleration multiple different ML workloads. Even though few accelerators provide innovative solutions, a significant bunch of accelerators either make incremental improvements and changes to the previously provided innovative solutions. With large organizations funding a major portion of ML accelerator research, the resulting products are proprietary leading to multiple groups reinventing the wheel. Moreover, software support for the architecture has been indispensable to success of the designed accelerators. Academic works which lack sophisticated software support albeit being replete with novel ideas rarely get adopted. Additionally, the privacy of the user data in the cloud accelerators and the security of these accelerators have not been extensively studied. Overall, the ML accelerator research space worked as individual groups spending time and money on similar solutions and lacked a community bridging industry and academia.

VI. Future

With processor and memory scaling being heavily limited by the limits of physics, and the inexorable growth in ML model sizes we are a point where we need innovative hardware and software (ML model) design to avoid hitting the accelerator wall [13]. Collaborations between academia and industry could aid the rapid adoption of privacy preserving machine learning solutions, spiking neural networks, and architecture solutions like embedded FPGAs, and newer process nodes. Software-Hardware co-design involving architects and ML researchers will exploit the best of both worlds. We also expect the adoption of Hardware Aware Neural Architecture Search techniques where a ML model with information about the underlying hardware, tunes the application's ML model to exploit the architecture. Finally, we also expect that the computing community will study the environmental impact of using power-hungry data centers for computation and come up with solutions which help preserve the world to reap the results of current technological advances.

VII. Conclusion

In this SoK study, we summarized the acceleration solutions currently present and then listed few open problems than needs to solved by the commmunity. Even though our study is not exhaustive, we have tried to cover a large sample space in-order to inform the reader about the large body work in this space.

References

[1] "Gap8 processor," Sep 2022. [Online]. Available: https://greenwaves-technologies.com/gap8_mcu_ai/

[2] F. I. Alexander Stanitzki, "Edge ai on low-footprint risc-v," January 2022. [Online]. Available: https://riscv.org/blog/2022/01/edge-ai-on-low-footprint-risc-v-alexander-stanitzki-fraunhofer-ims/

[3] A. Alvi and P. Kharya, "Using DeepSpeed and Megatron to train Megatron-Turing NLG 530B, the world's largest and most powerful generative language model," Oct 2021. [Online]. Available: https://www.microsoft.com/en-us/research/blog/using-deepspeed-and-megatron-to-train-megatron-turing-nlg-530b-the-worlds-largest-and-most-powerful-generative-language-model/

[4] J. Chen and X. Ran, "Deep learning with edge computing: A review," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1655–1674, 2019.

[5] Y. H. Chen, J. Emer, and V. Sze, "Eyeriss: A Spatial Architecture for Energy-Efficient Dataflow for Convolutional Neural Networks," in *Proceedings of the 43rd International Symposium on Computer Architecture*. Institute of Electrical and Electronics Engineers Inc., 8 2016, pp. 367–379.

[6] Y.-H. Chen, T.-J. Yang, J. Emer, and V. Sze, "Eyeriss v2: A Flexible Accelerator for Emerging Deep Neural Networks on Mobile Devices," 7 2018. [Online]. Available: http://arxiv.org/abs/1807.07928

[7] Y. Chen, T. Luo, S. Liu, S. Zhang, L. He, J. Wang, L. Li, T. Chen, Z. Xu, N. Sun, and O. Temam, "DaDianNao: A Machine-Learning Supercomputer," in *2014 47th Annual IEEE/ACM International Symposium on Microarchitecture*, vol. 2015-January. IEEE Computer Society, 1 2015, pp. 609–622.

[8] J. Choquette, "Nvidia Hopper GPU: Scaling Performance," in *2022 IEEE Hot Chips 34 Symposium (HCS)*, 2022, pp. 1–46.

[9] L. Chua, "Memristor-the missing circuit element," *IEEE Transactions on circuit theory*, vol. 18, no. 5, pp. 507–519, 1971.

[10] W. J. Dally, Y. Turakhia, and S. Han, "Domain-specific hardware accelerators," *Communications of the ACM*, vol. 63, pp. 48–57, 6 2020.

[11] J. Evans, "Nvidia Grace," in *2022 IEEE Hot Chips 34 Symposium (HCS)*, 2022, pp. 1–20.

[12] J. Fowers, K. Ovtcharov, M. Papamichael, T. Massengill, M. Liu, D. Lo, S. Alkalay, M. Haselman, L. Adams, M. Ghandi, S. Heil, P. Patel, A. Sapek, G. Weisz, L. Woods, S. Lanka, S. K. Reinhardt, A. M. Caulfield, E. S. Chung, and D. Burger, "A Configurable Cloud-Scale DNN Processor for Real-Time AI," in *Proceedings of the 45th Annual International Symposium on Computer Architecture*, ser. ISCA '18. IEEE Press, 2018, p. 1–14. [Online]. Available: https://doi.org/10.1109/ISCA.2018.00012

[13] A. Fuchs and D. Wentzlaff, "The Accelerator Wall: Limits of Chip Specialization," in *2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. Los Alamitos, CA, USA: IEEE Computer Society, Feb 2019, pp. 1–14. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/HPCA.2019.00023

[14] M. Horowitz, "1.1 computing's energy problem (and what we can do about it)," in *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, 2014, pp. 10–14.

[15] M. Horowitz, "Computing's energy problem (and what we can do about it)," in *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, 2014, pp. 10–14.

[16] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[17] D. Inc., "Deep Learning Inferencing with Mipsology using Xilinx ALVEO™ on Dell EMC Infrastructure," Oct 2019. [Online]. Available: https://dl.dell.com/manuals/common/deeplearning_mipsology_poweredge.pdf

[18] N. P. Jouppi, D. H. Yoon, M. Ashcraft, M. Gottscho, T. B. Jablin, G. Kurian, J. Laudon, S. Li, P. Ma, X. Ma, T. Norrie, N. Patil, S. Prasad, C. Young, Z. Zhou, and D. Patterson, "Ten Lessons from Three Generations Shaped Google's TPUv4i," in *Proceedings of the 48th Annual International Symposium on Computer Architecture*, ser. ISCA '21. IEEE Press, 2021, p. 1–14. [Online]. Available: https://doi.org/10.1109/ISCA52012.2021.00010

[19] N. P. Jouppi, D. H. Yoon, G. Kurian, S. Li, N. Patil, J. Laudon, C. Young, and D. Patterson, "A Domain-Specific Supercomputer for Training Deep Neural Networks," *Commun. ACM*, vol. 63, no. 7, p. 67–78, jun 2020. [Online]. Available: https://doi.org/10.1145/3360307

[20] N. P. Jouppi, C. Young, N. Patil, and D. Patterson, "A Domain-Specific Architecture for Deep Neural Networks," *Commun. ACM*, vol. 61, no. 9, p. 50–59, aug 2018. [Online]. Available: https://doi.org/10.1145/3154484

[21] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers, R. Boyle, P.-l. Cantin, C. Chao, C. Clark, J. Coriell, M. Daley, M. Dau, J. Dean, B. Gelb, T. V. Ghaemmaghami, R. Gottipati, W. Gulland, R. Hagmann, C. R. Ho, D. Hogberg, J. Hu, R. Hundt, D. Hurt, J. Ibarz, A. Jaffey, A. Jaworski, A. Kaplan, H. Khaitan, D. Killebrew, A. Koch, N. Kumar, S. Lacy, J. Laudon, J. Law, D. Le, C. Leary, Z. Liu, K. Lucke, A. Lundin, G. MacKean, A. Maggiore, M. Mahony, K. Miller, R. Nagarajan, R. Narayanaswami, R. Ni, K. Nix, T. Norrie, M. Omernick, N. Penukonda, A. Phelps, J. Ross, M. Ross, A. Salek, E. Samadiani, C. Severn, G. Sizikov, M. Snelham, J. Souter, D. Steinberg, A. Swing, M. Tan, G. Thorson, B. Tian, H. Toma, E. Tuttle, V. Vasudevan, R. Walter, W. Wang, E. Wilcox, and D. H. Yoon, "In-Datacenter Performance Analysis of a Tensor Processing Unit," in *Proceedings of the 44th Annual International Symposium on Computer Architecture*, ser. ISCA '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 1–12. [Online]. Available: https://doi.org/10.1145/3079856.3080246

[22] J. H. Kim, S.-h. Kang, S. Lee, H. Kim, W. Song, Y. Ro, S. Lee, D. Wang, H. Shin, B. Phuah, J. Choi, J. So, Y. Cho, J. Song, J. Choi, J. Cho, K. Sohn, Y. Sohn, K. Park, and N. S. Kim, "Aquabolt-XL: Samsung HBM2-PIM with in-memory processing for ML accelerators and beyond," in *2021 IEEE Hot Chips 33 Symposium (HCS)*, 2021, pp. 1–26.

[23] C. Lichtenau, A. Buyuktosunoglu, R. Bertran, P. Figuli, C. Jacobi, N. Papandreou, H. Pozidis, A. Saporito, A. Sica, and E. Tzortzatos, "AI Accelerator on IBM Telum Processor: Industrial Product," in *Proceedings of the 49th Annual International Symposium on Computer Architecture*, ser. ISCA '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 1012–1028. [Online]. Available: https://doi.org/10.1145/3470496.3533042

[24] S. Lie, "Cerebras Architecture Deep Dive: First Look Inside the HW/SW Co-Design for Deep Learning : Cerebras Systems," in *2022 IEEE Hot Chips 34 Symposium (HCS)*, 2022, pp. 1–34.

[25] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*. Springer, 2016, pp. 21–37.

[26] P. Mantovani, D. Giri, G. Di Guglielmo, L. Piccolboni, J. Zuckerman, E. G. Cota, M. Petracca, C. Pilato, and L. P. Carloni, "Agile soc development with open esp," in *2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*. IEEE, 2020, pp. 1–9.

[27] M. S. Murshed, C. Murphy, D. Hou, N. Khan, G. Ananthanarayanan, and F. Hussain, "Machine learning at the network edge: A survey," *ACM Computing Surveys (CSUR)*, vol. 54, no. 8, pp. 1–37, 2021.

[28] T. Norrie, N. Patil, D. H. Yoon, G. Kurian, S. Li, J. Laudon, C. Young, N. Jouppi, and D. Patterson, "The Design Process for Google's Training Chips: TPUv2 and TPUv3," *IEEE Micro*, vol. 41, no. 2, pp. 56–63, 2021.

[29] J. Park, M. Naumov, P. Basu, S. Deng, A. Kalaiah, D. S. Khudia, J. Law, P. Malani, A. Malevich, N. Satish, J. M. Pino, M. Schatz, A. Sidorov, V. Sivakumar, A. Tulloch, X. Wang, Y. Wu, H. Yuen, U. Diril, D. Dzhulgakov, K. M. Hazelwood, B. Jia, Y. Jia, L. Qiao, V. Rao, N. Rotem, S. Yoo, and M. Smelyanskiy, "Deep Learning Inference in Facebook Data Centers: Characterization, Performance Optimizations and Hardware Implications," *CoRR*, vol. abs/1811.09886, 2018. [Online]. Available: http://arxiv.org/abs/1811.09886

[30] R. Prabhakar and S. Jairath, "SambaNova SN10 RDU:Accelerating Software 2.0 with Dataflow," in *2021 IEEE Hot Chips 33 Symposium (HCS)*, 2021, pp. 1–37.

[31] B. Reagen, P. Whatmough, R. Adolf, S. Rama, H. Lee, S. K. Lee, J. M. Hernandez-Lobato, G. Y. Wei, and D. Brooks, "Minerva: Enabling

Low-Power, Highly-Accurate Deep Neural Network Accelerators," in *Proceedings of the 43rd International Symposium on Computer Architecture*. Institute of Electrical and Electronics Engineers Inc., 8 2016, pp. 267–278.

[32] A. Reuther, P. Michaleas, M. Jones, V. Gadepally, S. Samsi, and J. Kepner, "AI and ML Accelerator Survey and Trends," in *2022 IEEE High Performance Extreme Computing Conference (HPEC)*, 2022, pp. 1–10.

[33] A. Samajdar, Y. Zhu, P. Whatmough, M. Mattina, and T. Krishna, "Scale-sim: Systolic cnn accelerator simulator," *arXiv preprint arXiv:1811.02883*, 2018.

[34] K. Sankaralingam, T. Nowatzki, V. Gangadhar, P. Shah, M. Davies, W. Galliher, Z. Guo, J. Khare, D. Vijay, P. Palamuttam, M. Punde, A. Tan, V. Thiruvengadam, R. Wang, and S. Xu, "The Mozart Reuse Exposed Dataflow Processor for AI and beyond: Industrial Product," in *Proceedings of the 49th Annual International Symposium on Computer Architecture*, ser. ISCA '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 978–992. [Online]. Available: https://doi.org/10.1145/3470496.3533040

[35] Y. S. Shao, B. Reagen, G.-Y. Wei, and D. Brooks, "Aladdin: A pre-rtl, power-performance accelerator simulator enabling large design space exploration of customized architectures," in *2014 ACM/IEEE 41st International Symposium on Computer Architecture (ISCA)*, 2014, pp. 97–108.

[36] H. S. Stone, "A logic-in-memory computer," *IEEE Transactions on Computers*, vol. 100, no. 1, pp. 73–78, 1970.

[37] V. Sze, Y.-H. Chen, J. Emer, A. Suleiman, and Z. Zhang, "Hardware for machine learning: Challenges and opportunities," in *2017 IEEE Custom Integrated Circuits Conference (CICC)*. IEEE, apr 2017. [Online]. Available: https://doi.org/10.1109%2Fcicc.2017.7993626

[38] E. Talpes, D. Williams, and D. D. Sarma, "Dojo: The microarchitecture of tesla's exa-scale computer," in *2022 IEEE Hot Chips 34 Symposium (HCS)*, 2022, pp. 1–28.

[39] V. Verma, "Ai-risc: Scalable risc-v processor for iot edge ai applications," Ph.D. dissertation, University of Virginia, 2022.

[40] T.-J. Yang, Y.-H. Chen, and V. Sze, "Designing energy-efficient convolutional neural networks using energy-aware pruning," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 6071–6079.