Survey On Processor Design For Neural Network Accelerators

By Hayden Gemeinhardt

Abstract

As with most technical problems, there are two categories of hurdles in the research of artificial intelligence (AI) and machine learning (ML). The software and the hardware. Pre-2000s, AI and ML were held back by the technology of the time. In the last couple of decades, hardware has finally caught up to enable us to begin research and produce tangible results. However, as our knowledge of AI increases seemingly exponentially, so does our need of hardware, and hardware is quickly becoming the main hurdle once more.

The rate of our knowledge growth on the software side of things far exceeds that of the progress in hardware advancements. Thus, given enough time, hardware will full circle back to slowing down research. We can still pack a room full of thousands of CPUS, GPUS, and the like, but it is very costly.

This survey will look at some of the current innovations in hardware addressing the performance, power consumption, and other aspects of computing being made in response to the growing need of better hardware.

Survey of Processors

Artificial Intelligence is an extremely diverse field. There is AI for facial recognition, spam detection, risk assessment, targeted marketing, robot navigation, inventory management, you name it. As such, there is a variety of different processors for these different uses.

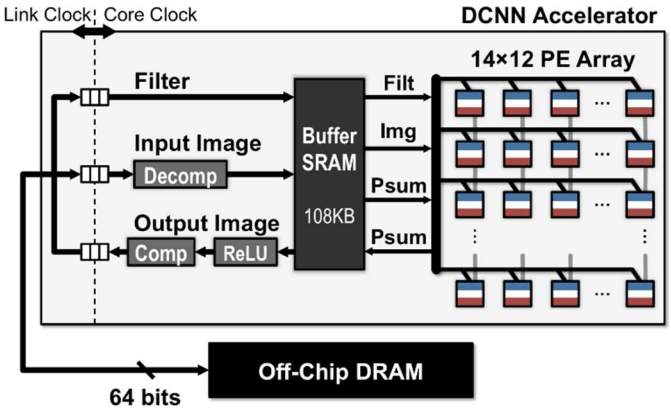
For example, a university or company doing research may not need the same types of chips a data center uses. These research chips typically consume a considerable amount of less power as compared to the data center chips, though lose some performance in the process. One of the main reasons for this is data centers are more focused on inference for predictions versus research chips used for training.

Processors With Low Power Usage

A. MIT's Eyeriss Chip [1, 2]

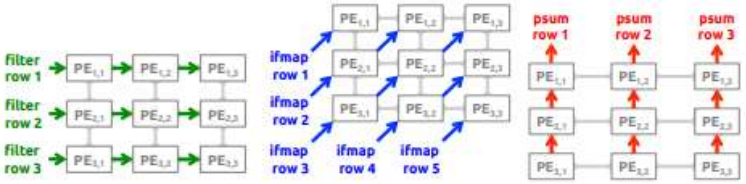
Eyeriss is an accelerator for convolutional neural networks (CNNs), optimizing energy efficiency on a reconfigurable board. Their main innovation was a processing dataflow called row stationary (RS). They realized that due to a large amount of data required for computation, there was an inefficiency in the data movement from on-chip to off-chip. The data transfer is more energy-consuming than the computation itself. Thus, RS was created. RS is designed to maximize reusing data locally on the chip to reduce the amount of data movement such as DRAM access.

The chip features a 14x12 processing element (PE) array, or about 168 PEs. These PEs are where RS comes into play.



Eyeriss breaks higher-dimensional convolutions into one-dimensional "primitives" that run in parallel, input coming from the off-chip DRAM. Each primitive runs on one row of filter weights and ifmap pixels, and generates one row of psums. Then psums from other primitives are grouped together to generate the ofmap pixels.

Now, each primitive is mapped to a single PE to be processed, so each computation of rows stays within that PE, hence the name row stationary. Thus, the filter weights and ifmap pixels can be reused within the register files of each PE.



To help visualize how mapping a primitive to a set of PEs works for a two-dimensional convolution, take this representation. Filter weights are reused across PEs horizontally, ifmap pixels are reused across PEs diagonally, and the psums are accumulated across PEs vertically. This is known as the logical mapping.

Then, there is the second step, the physical mapping. The number of convolution layers is equal to the number of filter channels (C) times the number of filters (M) times the batch size (N). Thus, this would require CxMxN PE sets.

They use a special technique "folding", in which they map multiple primitives from different PEs onto a single PE. This is because the same filter weights can be reused on N sets (fmap batch size), the same ifmap pixels on M sets (number of filters), and the same psums across C sets (channels). Basically, this leaves a lot of reusability on the separate PEs

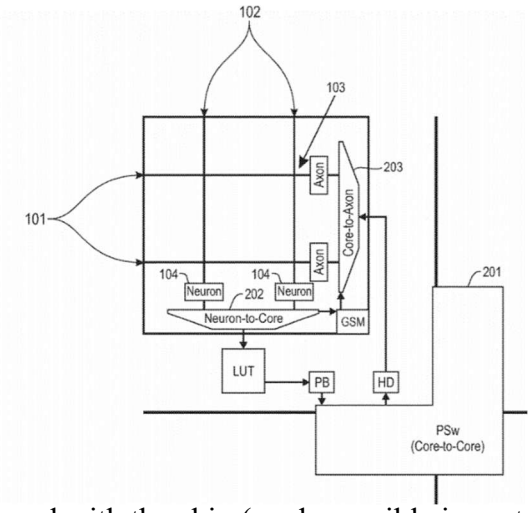
The amount of folding that occurs depends on two factors: the register file size of each PE and the PE array size. Beyond that it becomes solely an optimization issue.

B. IBM's TrueNorth Chip [1, 3]

Funded by DARPA's SyNAPSE program, the TrueNorth chip aims to emulate human neurological activity, largely in part to study how our brains do computation so efficiently.

One of the key elements in TrueNorth was to implement asynchronous circuits. Neurons in the brain activate without a synchronous clock, and given TrueNorth mimics the brain, it only made sense to make it asynchronous. However, the chip is also extremely power efficient- only

using up to 275 mW of power, also surprisingly similar to the brain that uses just 20 watts of power. Part of this is due to the use of digital spiking neural networks.



All communications on the chip occur with "spikes", similar to how neurons communicate. In the figure, you can see there are two main parts, the neuron-to-core as output and core-to-axon as input. When a neuron spikes, it send a message to an axon on another core.

This reduces the factors affecting the processing of information to two main aspects: the timing of spikes and the identity of the synapses used (which neurons are connected, are they inhibitory, etc.). Optimizing this leads to the incredible efficiency found with the chip (made possible in part by the reconfigurable nature of the chip).

One of the main methods of optimization is through a custom made software used to create efficient core placement to minimize the communication between the 4096 cores across all chips and instead maximize the communication between cores on the same chip. This is similar to the Eyeriss chip that focused on maintaining as much processing on the same PE as possible, since data transfer is costly.

Cellphone / Mobile GPU-based Neural Engines

Modern smartphones are starting to implement "AI chips" on board. Previously, this was done in two ways. One was over the cloud. However, this was extremely inefficient and could not be used without internet/data access. The other was using utilizing the on board GPU, which used a lot of power resulting in a less than ideal battery life.

A. Huawei Kirin 980 [1, 4]

The Kirin 970 was in fact the first mobile chipset to incorporate a dedicated Neural-Network Processing Unit (NPU). This allowed it to use AI without internet connection and allow the phone to recognize the differences between say a dog and a cat. The next generation Kirin 980 incorporates two NPUs, essentially doubling down on the design.

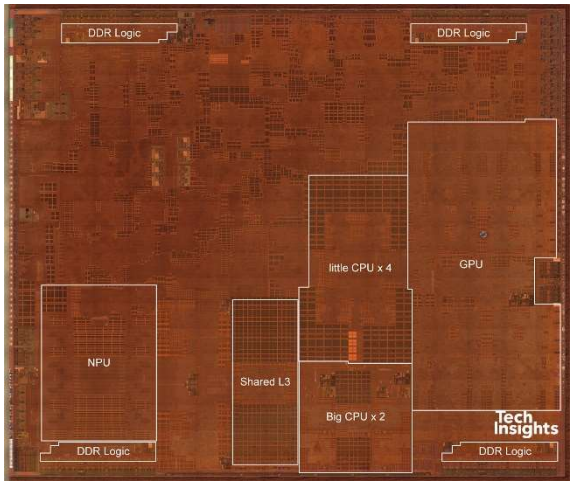
An NPU is designed specifically for neural network tasks, meaning it is much more efficient than onboard CPUs and GPUs for things like facial or image recognition, something becoming increasingly more common in modern mobile technology.

B. Apple's A12 Processor [1, 5]

As mentioned previously, AI envelopes tasks such as facial recognition, something used quite frequently on phones. As such, Apple's new processors use "Bionic" processors.

Compared to the Kirin 980, the A12 is much more efficient, which is not surprising considering Apple's obsession over having state-of-the-art technology in all of their newest phones to stay ahead of the competition.

A key element of this chip is its flexibility between full power and idleness. It utilizes two different types of processors to achieve this, four "LITTLE" processors, designed for maximum power efficiency, and two "Big" processors, designed for maximum computational performance. Mirroring this comes the GPU, capable of parallelizing convolution operations.



When the phone is required to perform a more complicated task like facial recognition, the neural engine Bionic kicks in and surges the power utilization above the usual maximum for a short period of time for fast inference. This has brought a large increase in battery life to iPhones and made neural-network tasks much faster.

Data Center Systems [6]

Almost everything online today runs through a data center. In fact, throughout our day, we may be interacting with many more data centers than we realize. And at the core of it all lies the chips and cards that run it.

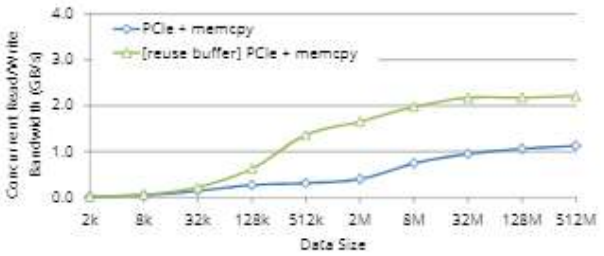
Every time you get an advertisement, it has likely gone through a targeted marketing program run by AI. Or when you buy something online, those inventories are in part run by AI. Auto generated search results are influenced by AI's decisions. Not only do these data centers have to uphold the traffic running through them, but often also have to process all of the AI-driven programs influencing a user's experience.

The hardware driving this includes CPUs, GPUs, CPU-controlled FPGAs, and dataflow accelerators. GPUs have proven to be more effective at computation-heavy tasks like machine learning as compared to CPUs, but FPGAs have proven to be even more beneficial when used properly. And the next step up from that is utilizing CPUs to accelerate the FPGAs.

The CPU-controlled FPGAs are the most interesting of the bunch, so let's focus on that. This combo is focused around a CPU used to rapidly download FPGA configurations onto the board, and then send operations to the FPGA for processing.

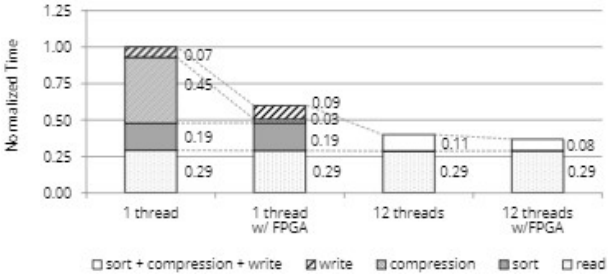
In order to utilize this combination, a few considerations need to be made for an optimal dataflow model. Following along this study which looks at doing so by combining data-level parallelism on a multicore CPU, hardware specialization on an FPGA, and a pipeline parallelism between the multiple cores and the FPGA.

First they looked at the bandwidth for each communication between the CPU and FPGA. The study found first that concurrent read/write bandwidth increases as the data size increased. Secondly, reusing memory objects helps increase the bandwidth (a common theme found through out this survey).



Another consideration is the inmemory sorting. The first step of in-memory sorting is reading the data into memory. Next, the parallel threads are used to sort chunks of data, compress it, and write the data to a file.

In the figure to the right, we see that more threads and an FPGA accelerator increase the runtime speed for sorting. Note for the two columns using 12 threads, sort and compress stages are already optimized, and it is simply the read stage that experiences an 8% improvement with the FPGA accelerator.



The third consideration the study made was runtime thread allocation. There were two different scenarios: parallel read and a dataflow implementation of Samtool sorting. By partitioning input SAM files into smaller files, the read stage can be parallelized (and consequently imporving SSD performance by threefold). On the other hand, the dataflow model partitions the application into read, sort, and compress+write stages and manages input/output data between these stages using queues.

Then, they add on two more scenarios, with or without an FPGA accelerator.

Without an FPGA accelerator, the parallel read outperforms the dataflow model since the dataflow model uses queues, taking up extra memory. However, with the FPGA accelerator, the dataflow model is faster since it can execute computation intensive tasks and I/O intensive tasks simultateously while parallel read executes in a stage-by-stage manner. In other words, the read stage is parallelized using 6 threads and then sort+compression+write stage uses 12 threads, while the dataflow model can allocate threads dynamically to achieve the simultaneous tasks.

Conclusion

In this survey, we looked at three different categories of hardware accelerators.

First, chips with optimal power usage. We saw MIT's Eyeriss chip using Row Stationary and IBM's TrueNorth chip mimicing neurological activity in the human brain using Nuerons, Axons, and a spiking neural network.

Second, mobile chips implementing Neural-Network Processing Units. Huawei first introduced this in their Kirin 970 chip and Apple improved upon that innovation in their A12 processor with power efficiency between idle and full loads. The A12 used little-big processors and power "surges" for fast NPU computations.

Thirdly, we investigated CPU-FPGA designs used in data centers. The study used focused on optimizing coordination between the two to further utilize the lower power and high performance created by combining the two.

Sources

https://arxiv.org/abs/1908.11348

[1] Reuther, Albert, et al. “Survey and Benchmarking of Machine Learning Accelerators.” *ArXiv.org*, 29 Aug. 2019, arxiv.org/abs/1908.11348.

https://ieeexplore.ieee.org/document/7551407

[2] Y. Chen, J. Emer and V. Sze, "Eyeriss: A Spatial Architecture for Energy-Efficient Dataflow for Convolutional Neural Networks," *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, Seoul, 2016, pp. 367-379, doi: 10.1109/ISCA.2016.40.

https://www.nextplatform.com/2018/09/27/a-rare-peek-into-ibms-true-north-neuromorphic-chip/

[3] Hemsoth, Nicole, and Michael Feldman. “A Rare Peek into IBM's True North Neuromorphic Chip.” The Next Platform, 27 Sept. 2018, www.nextplatform.com/2018/09/27/a-rare-peek-into-ibms-true-north-neuromorphic-chip/.

https://www.researchgate.net/publication/336890831_Huawei_GPU_Turbo_Technology_Research

[4] Zhang, Ningyuan. (2019). Huawei GPU Turbo Technology Research.

https://www.researchgate.net/publication/325053646_A_Microarchitectural_Study_on_Apple's_A11_Bionic_Processor

[5] Banerjee, Debrath. (2018). A Microarchitectural Study on Apple's A11 Bionic Processor.

https://www.researchgate.net/publication/317623875_CPU-FPGA_Co-Scheduling_for_Big_Data_Applications

[6] Cong, Jason & Fang, Zhenman & Huang, Muhuan & Wang, Libo & Wu, Di. (2017). CPU-FPGA Co-Scheduling for Big Data Applications. To appear in IEEE Design and Test 2017. PP. 10.1109/MDAT.2017.2741459.