# Survey on State of the Art Machine Learning Accelerator Design

Victor Martin Agostinelli III, Indhu Priya Karunagaran,
Diksha Pritam Todankar, and Kazi Ahmed Asif Fuad

[Mid-report] February 2022

## 1 Introduction

Machine learning solutions are increasingly popular and applicable, but with the slowing down of Moore's law comes a necessity for dedicated hardware to accelerate those solutions further. Machine learning accelerators are, as such, a hot area of both academic and commercial interest. Companies like Tesla and Google are using dedicated hardware to enable tasks such as real-time vision for autonomous vehicles or translation between disparate languages. In parallel, academic groups across the world are focused on exploring the relatively young design space for these accelerators. Developments in this space are constant, exciting, and have resulted in designs that operate quite differently from one another, in many cases with very clear trade-offs between them in terms of power consumed, area taken up, and speed.

This survey seeks to acquire a holistic view of the design space, some of the techniques that state of the art accelerators are being developed with, and a sense of which modern architectures are relevant and how they compare with one another. To this end, a number of papers published in relevant machine learning journals and at specific conferences must be engaged with alongside notable commercial options. Indeed, this survey will be limited to those accelerator architectures being developed by academically focused groups, as vastly more detailed literature exists on those designs and the process that went into creating them. Several accelerator designs have already been identified as particularly promising, including Eyeriss from the Massachusetts Institute of Technology [5], FlexFlow from the Chinese Academy of Sciences [2], and MAERI from the Georgia Institute of Technology [4], and will be included in this survey's comparisons. In addition to this, some limited simulation lies within the scope of this survey, extracting performance and efficiency details from these designs using cutting edge accelerator evaluation tools [6, 7].

## 2 Accelerator Design Background

A few key concepts should be kept in mind that apply to most, if not all, machine learning accelerators, including the typical components of a machine learning accelerator, the design stages for these accelerators, and how useful metrics can be extracted from a given piece of dedicated hardware that allow for meaningful comparisons to other architectures.

As far as components go, while this can and does vary on a case to case basis, the bulk of most accelerators is typically dedicated to a systolic array of processing engines or processing elements (PEs). These PEs are capable of accelerating typical arithmetic operations that a specific accelerator might need to often deal with, with one incredibly common example being dedicated hardware within a PE for multiply-and-accumulate operations (MACs). Tremendous energy efficiency gains can be acquired in the careful use of the array of PEs in a given accelerator, making their efficient arithmetic for some model or task extremely important [1]. In addition, many accelerators feature on-board buffers for less costly memory accesses, when necessary. These buffers allow for a given accelerator to reuse specific pieces of data when it would be too costly in terms of time or energy efficiency to pull it from off-chip memory.

The traditional design flow for a machine learning accelerator typically follows as such: the specification of a dataflow, the selection of a mapping strategy, the allocation of hardware resources [6]. The dataflow for a given accelerator usually defines the accelerator's data reuse strategy in addition to defining some strategies for high PE array utilization that are beyond the scope of this survey [6]. Choosing a dataflow

is typically done either through heuristics, brute force search of a very limited design space, or selection by experts [6]. The mapping strategy details how resources are mapped when a given machine learning model demands more PEs or buffer space than the hardware allows, which can be common for large networks or accelerators on the edge [7]. It is often wrapped together with the process of hardware resource allocation, as its design space is quite small. In line with that, hardware resource allocation is, ultimately, the partitioning of available buffer space and PEs according to the model's computational needs and is accomplished through DRL-based methods when it comes to cutting edge solutions, although such developments are still nascent [7].

When attempting to extract useful metrics, the first set of metrics that appear as fairly universal for machine learning accelerators would be power efficiency, performance for a given model and benchmark, and physical size [6]. The final metric is a simple one to measure and make comparisons with, but the first two can be more difficult to extract, especially if a team of architects is attempting to decide between suitable accelerator architectures before design-time and must simulate. Fortunately, a few academic groups have worked to provide tools that are capable of estimating one or both of these metrics, including MAESTRO [6], a novel tool developed by a group at the Georgia Institute of Technology.

# 3 Literature Review

## 3.1 Energy-Efficient Convolution Architecture Based on Rescheduled Dataflow

Rescheduled dataflow convolution has its hardware architecture which enhance energy efficiency. For involving a large amount of memory access and computation, previous accelerators use parallel processing elements to overcome real-time constraints. Convolution neural networks (CNN) have been used to various number of applications due to unprecedented accuracy and particularly in object recognition as it can discriminate hundreds of object. Here we understand more about 1D and 2D convolution operations and on-chip memory access patterns of two state-of the-art CNN accelerators. Then we analyze and renovate the dataflow of convolution and summarize the most suitable CNN architecture which can maximize the energy efficiency by reducing redundant on-chip memory access. Since the filtering operation is repeated hierarchically ,the CNN achieves a high recognition rate. According to this filtering, the convolution operation makes various effects on the image input such as blurring,sharpening and edge detection. By using accumulation,combination and convolution products we can achieve an output image. The process that takes place is the accelerators employs parallel multipliers to compute relevant products to an output cycle. The adder tree accumulates the product to a fixed number to support limited convolution operations. By the row-stationary dataflow structure, each PE computes partial sum relevant to results and row of weights aggregated by passing to the upper PEs. The rescheduled dependence dataflow comes into picture to reduce the energy overhead consumed by on-chip memory access. Few changes are made to reschedule the data flow. The number of steps is directly proportional to the size of the input feature by using OM. By accumulating the partial sums incrementally has the input data loaded only once from IN, which enhance the overall energy efficiency. Output features can be calculated by gradually accessing IM only once and computing the products related to the input at once.We get to know that the significant portion of the energy is mainly taken in on-chip memory access,hence the proposed accelerator adopting this rescheduled dataflow is effective in enhancing the energy efficiency.

## 3.2 FlexFlow

Authors of FlexFlow [2], Wenyan Lu *et al.* proposed a flexible dataflow architecture capable of mitigating the mismatch between the varying dominant parallel types of CNN and computing engine supported parallel types. Three prominent architectures: Systolic, 2D-Mapping, and Tiling support synapse parallelism, neuron parallelism, and feature map parallelism receptively. However, these architectures fail to explore parallelisms efficiently as the number of feature maps, the number of output feature maps, the size of output feature map and the size of kernel varies dramatically in different layers of CNNs. Although FlexFlow implementation requires slightly larger area, the computing resource utilization of FlexFlow is 80% compared to the baseline architectures' max 60% utilization for mixture of workloads. It achieves 2x performance and power efficiency speed up compared to Systolic and 2D Mapping whereas 10x for Tiling while maintaining high scalability.

### 3.3 ShiDianNao

Yet another group focused on highly efficient accelerator designs, the authors of ShiDianNao [8] honed in on an architecture specializing in efficient image recognition, capable of being implemented on the edge alongside a sensor array. Correspondingly, the models that they were capable of supporting were generally much more compact, with far fewer parameters expected and a requirement of all memory requests being answered on-chip as opposed to allowing access to some off-chip memory. Especially in regards to image processing on the edge, the authors justified expected drops in computational performance by considering sensor bandwidth and throughput to be a limiting factor, thus rendering larger computational blocks unnecessary [8].

The comparison of ShiDianNao to other pieces of hardware capable of running the aforementioned DNN and CNN models that this group was focused on was somewhat lacking, unfortunately. They compared this architecture primarily to one of their previous designs, DianNao [11], in addition to some general purpose hardware and found, unsurprisingly, that when normalized compared to the resources their previous design had, ShiDianNao outperformed both comparable general purpose hardware and their previous design on nearly all workloads, failing to its predecessor only where raw throughput was concerned [8]. Energy savings abounded with this design, as expected.

### 3.4 Eyeriss

While a number of CNN accelerator architectures have sought out extreme energy efficiency, Eyeriss remains interesting as the first to propose a rather novel approach: making use of a row-stationary dataflow [5]. Whereas previous architectures traditionally focused on three dataflows built on minimizing weight, input, or output accesses for CNN accelerators, a row-stationary dataflow differs in that it approaches tasks adaptively based on data reuse opportunities and hardware allocation for a given model and its layers. This dataflow depends on an efficient mapping process, however, for individual computations across the systolic array of PEs. The research group at MIT that built Eyeriss, in collaboration with NVIDIA, focused on breaking down the convolutional layers of a given CNN model into what they call "1D convolution primitives" that they then mapped over the course of a two-step process [5]. The first of those steps involved mapping the convolution primitives to a logical array of PEs, with that logical array typically being significantly larger than what is physically available in hardware. Following efficient logical mapping, physical mapping was necessary, taking logical PEs and scheduling their tasks on physical PEs such that, oftentimes, a single physical PE would execute the computations of a "row" of logical PEs that would handle a given 1D convolution primitive.

The evaluation of the Eyeriss architecture and its row-stationary dataflow was focused on comparisons to other dataflow options for CNN accelerators, featuring the more traditional weight, input, and output stationary dataflows in this case, on AlexNet, a CNN that was popular during its inception for accurate object recognition in computer vision tasks [9] and with consistent hardware and parallelism opportunities. Across the board, Eyeriss and its dataflow selection was found to access DRAM less often (meaning that less energy and time was wasted on costly memory acesses) and use less energy per operation than its competitors [5].

### 3.5 MAERI

In contrast to a number of other designs, MAERI is focused on flexibility above all. Whereas CNN accelerators often have to select a single dataflow to specialize in, MAERI was built with reconfigurable interconnects, meaning a flexible network on chip (NoC), between its PEs, allowing for a flexible transition of data across the array [4]. Interestingly, this adaptability allows for mapping across highly irregular models. Should optimization for a given model require a non-traditional and unusually structured dataflow, MAERI is capable of supporting such a network. While this flexibility is important in its own right, aiding multi-purpose accelerator design, the team behind MAERI was also focused on underlining that the overhead for MAERI's NoC was minimal and that it performed competitively compared to other options.

The team behind MAERI evaluated the architecture by comparing it specifically to Eyeriss and even more traditional systolic array structures. They found that, generally, MAERI was a more compact architecture as its PEs tended to require fewer elements than those of Eyeriss (a simpler RF structure, separation between types of PEs for adder switches, multipler switches, simple switches, etc.). Holding the number of PEs constant or the area constant, MAERI proved to hold advantages over Eyeriss related primarily to latency,

as MAERI is relatively power-hungry. The team behind MAERI attributes this primarily to the design's very high levels of utilization, as their mapping of a given model can be relatively irregular in comparison to Eyeriss and other designs.

### 3.6 ConfuciuX and MAESTRO

Deep Neural Network accelerator architecture designs are dominated by two main components, dataflow style and allocated total hardware resources. Dataflow utilizes reuse of activations, weights, and outputs in DNNs by employing computation order, parallelization order, and tiling strategies. Usual hardware resources are on-chip processing elements and memories. It is possible to allocate hardware resources in many ways to optimize a particular dataflow architecture of a given DNN model. Recent research shows, an optimized HW resource allocation is more important for DNN performance that it's dataflow architecture [10, 7]. Authors of ConfuciuX [7], Sheng-Chun Kao et al. developed an autonomous method to allocate hardware resources for a given model and dataflow by employing a combination of reinforcement learning method, REINFORCE for global search and genetic algorithm for fine tuning. ConfuciuX takes a DNN model, the deployment scenario, the optimizing objective, and the platform constraints and it generates an optimized hardware allocation. Based on training, the RL agent continuously generates "actions" which are evaluated by an analytical model such as MAESTRO [6], an environment which outputs "rewards" trains underlying policy network to maximize the "reward". Later, platform constraints such as area or power is assigned to environment to punish actions if those violate the constraints. This proposed method achieves optimized hardware configuration 4.7 to 24 times faster than existing other techniques at that time [7].

MAESTRO [6] is a systematic data reuse based analytical cost model that provides more than 20 statistical metrics to evaluate DNN accelerator design space. It was validated with cycle accurate RTL simulation for MAERI [4] and Eyeriss [5] accelerators and results were within 3.9% absolute error. Moreover, it provides fast estimation, taking only 493 ms to analyze the layers7 of Resent50 on a 256PE NVDLA style accelerator with Core-i9 CPU and 16 GB RAM.

## 4 Machine Learning Accelerator Architecture and Design Techniques

## 5 Simulation Methodology

## 6 Analysis of Results

## 7 Discussion

## References


[1] Y. Chen, J. Emer and V. Sze, "Using Dataflow to Optimize Energy Efficiency of Deep Neural Network Accelerators," in IEEE Micro, vol. 37, no. 3, pp. 12-21, 2017, doi: 10.1109/MM.2017.54.

[2] W. Lu, G. Yan, J. Li, S. Gong, Y. Han and X. Li, "FlexFlow: A Flexible Dataflow Accelerator Architecture for Convolutional Neural Networks," 2017 IEEE International Symposium on High Performance Computer Architecture (HPCA), Austin, TX, 2017, pp. 553-564, doi: 10.1109/HPCA.2017.29.

[3] J. Jo, S. Kim and I. Park, "Energy-Efficient Convolution Architecture Based on Rescheduled Dataflow," in IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 65, no. 12, pp. 4196-4207, Dec. 2018, doi: 10.1109/TCSI.2018.2840092.

[4] Hyoukjun Kwon, Ananda Samajdar, and Tushar Krishna. 2018. MAERI: Enabling Flexible Dataflow Mapping over DNN Accelerators via Reconfigurable Interconnects. SIGPLAN Not. 53, 2 (February 2018), 461–475. DOI:https://doi.org/10.1145/3296957.3173176.

[5] Yu-Hsin Chen, Joel Emer, and Vivienne Sze. 2016. Eyeriss: a spatial architecture for energy-efficient dataflow for convolutional neural networks. SIGARCH Comput. Archit. News 44, 3 (June 2016), 367–379. DOI:https://doi.org/10.1145/3007787.3001177.

[6] H. Kwon, P. Chatarasi, V. Sarkar, T. Krishna, M. Pellauer and A. Parashar, "MAESTRO: A Data-Centric Approach to Understand Reuse, Performance, and Hardware Cost of DNN Mappings," in IEEE Micro, vol. 40, no. 3, pp. 20-29, 1 May-June 2020, doi: 10.1109/MM.2020.2985963.

[7] S. -C. Kao, G. Jeong and T. Krishna, "ConfuciuX: Autonomous Hardware Resource Assignment for DNN Accelerators using Reinforcement Learning," 2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), 2020, pp. 622-636, doi: 10.1109/MICRO50266.2020.00058.

[8] Z. Du, R. Fasthuber, T. Chen, P. Ienne, L. Li, T. Luo, X. Feng, Y. Chen, and O. Temam, "ShiDianNao: Shifting Vision Processing Closer to the Sensor," in ISCA, 2015.

[9] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in IEEE CVPR, 2016.

[10] X. Yang, M. Gao, J. Pu, A. Nayak, Q. Liu, S. E. Bell, J. O. Setter, K. Cao, H. Ha, C. Kozyrakis et al., "DNN dataflow choice is overrated," arXiv preprint arXiv:1809.04070, 2018.

[11] T. Chen, Z. Du, N. Sun, J. Wang, and C. Wu, "DianNao: a small-footprint high-throughput accelerator for ubiquitous machinelearning," in Proceedings of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), Salt Lake City, UT, USA, 2014, pp. 269–284.