Overview of the State of the Art in Embedded Machine Learning

Liliana Andrade, Adrien Prost-Boucle and Frédéric Pétrot
Univ. Grenoble Alpes, CNRS, Grenoble INP†, TIMA, F-38000 Grenoble, France
name.surname@univ-grenoble-alpes.fr

***Abstract*—Nowadays, the main challenges in embedded machine learning are related to artificial neural networks. Inspired by the biological neural networks, artificial neural networks are able to solve complex problems, by performing a tremendous amount of relatively simple parallel computations. Embedding such networks in autonomous devices raises the issues of energy efficiency, resource usage and accuracy.**

The aim of this paper is to provide a comprehensive analysis of the efforts made in recent years to implement artificial neural network architectures suitable for embedded applications.

I. Introduction

Embedded applications cover many needs, among which three are quite representative: (1) ultra-low power sensors that are supposed to last for weeks or months on battery power, (2) mobile phones, that have stringent power constraints while being supposed to do what a desktop computer does and more, and (3) intelligent vehicles, that have big power sources, but are supposed to use them to travel, not compute. Clearly, these application classes have different timing, power and price constraints, and therefore our overview on artificial neural networks (ANN) will outline the spectrum of solutions currently under investigation.

At first, we shall note that the work of Hinton *et al.* [1] in 2006 on deep learning [2] resurrected the ANN field. Indeed, by inventing a proper way to train very deep neural networks (DNN), this work made possible to realize many artificial intelligence (AI) tasks more efficiently than by using *ad-hoc* algorithms, and a turning point occurred in 2012 when Krizhevsky won the ImageNet Large Scale Visual Recognition Challenge [3]. It comes at no surprise that deep learning induces many neuron layers, and therefore *inference*, *i.e.* the task of using a pretrained network to solve an AI problem, requires a lot of computation. However, the capability of obtaining very high accuracy using very complex networks also raised the interest of using ANN for less complex tasks, thanks to their generality and flexibility.

Depending on the context, current practical implementations make use of either general purpose processors, for small, low throughput applications, or of specialized processors, typically a GPU using 16 bit floating point arithmetic, or a specific DSP-based hardware using fixed or floating point numbers when high performances are required. Fig. 1 illustrates the concept of Von Neumann architecture, in which the processor and the memory are far apart. Efficiently implementing inference on this kind of architecture is difficult, because first this workload consists only of multiply-accumulate operations (MAC) and when used solely for that purpose the processor is largely under-used, and second the access to the memory is sub-optimal as it goes through a shared bus, and then caches, while given the ANN behavior the content of the cache is not reused.

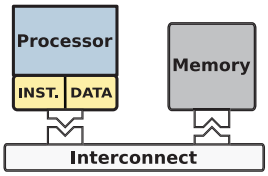

Fig. 1. Textbook Von Neumann architecture.

As a consequence, the hardware community soon realized that specialized accelerators had the potential of being much more efficient than processors for this kind of computation [4], and the hardware acceleration of inference has been since the focus of many research works. But not only is the computation demanding, with tens of layers and thousands of neurons per layer, so is also storage to hold the weights and fetch them with the appropriate throughput for the neurons. To give an idea of the magnitude at hand, VGG [5], the runner-up of the 2014 ILSVRC contest, requires 130 M parameters. Inception v4 [6], very deep unconventional CNN reaching top accuracy, requires 35 M parameters. More than the computation itself, the access to these parameters becomes the bottleneck of the dedicated architectures because of the power consumption they induce. Indeed, according to [7], for a given 45 nm technology, the energy per operation of an off-chip DRAM access is $3500\times$ higher than that of a 16-bit addition, and around 60 to $80\times$ higher than an on-chip SRAM access. Given these numbers, the access to the weights from external memory largely exceeds the needs of the computations to be performed by the neurons, both in timing and power consumption.

To bring the weights close to the computation, specialized architectures tend to integrate many memory cuts with fully dedicated processing elements. By recognizing the central role of memory in the inference process, architectures based on the template illustrated Fig. 2 have a much higher throughput and power efficiency. IBM TrueNorth chip, whose neuron plus synaptic weights memory is illustrated Fig. 3, is representative of that type. Recent FPGAs also fit this template very well, as they contain many memory blocks (up to 4320 18Kb blocks in

†Institute of Engineering Univ. Grenoble Alpes

978-3-9819263-0-9/DATE18/©2018 EDAA

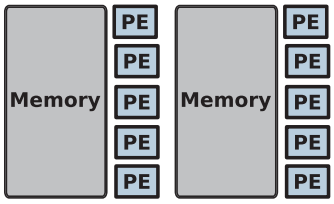

Fig. 2. Memory centric architecture.

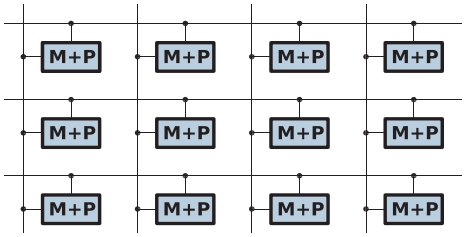

Fig. 4. Memory with integrated processing elements.

Xilinx's Kintex family, and up to 7033 20Kb blocks in Intel's Stratix 10 family).

A more forward looking approach, not mature yet but subject of a lot of trials and discussion, consists of integrating the computation into the memory itself, as shown Fig. 4. Most of these approaches use non-conventional memories and among them some use spiking network architectures, betting on the fact that being event-based, they will consume less power.

Unlike [9] that references a mere 2682 research papers in hardware implementation of neural networks, we by no mean aim at completeness in the present paper. Instead, we discuss the major contributions presented the last years in terms of the four alternative paths mentioned above, limiting ourselves to the most applicable approaches for embedded machine learning.

II. Low-Power, Low-Area Digital Computations

Floating-point data is what usual training frameworks use and produce, hence inference is generally done using floating-point operations. As floating-point capable operators are very area and power consuming, several approaches have been proposed to avoid them.

Benefiting from floating point representation while avoiding floating point multiplication is proposed in LookNN [10]. The idea is to constraint the weights in such a way that all possible multiplications can be tabulated in a TCAM-based lookup table. Using a weight clustering procedure and a specific retraining phase, and running on memristor-based technology, LookNN claims an average of $2.2\times$ energy improvement and a $2.5\times$ speedup without loss of accuracy as compared to a GPU implementation.

Venkatesh *et al.* [11] proposed to keep floating-point accumulations but to use only balanced ternary weights (possible values are -1, 0 and $+1$). This approach removes the need for multipliers and only requires add/sub operators, which brings notable area and power savings.

Another major class of embedded solutions favors integer-based inference approaches. It has been shown that using 16-bit or even 8-bit quantization has very low or no impact on accuracy when using direct quantization of traditional floating-point training results [12]. Going below entails excessive accuracy drop but this can be compensated by quantization-aware retraining.

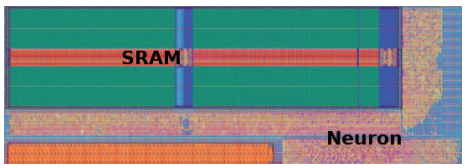

Fig. 3. IBM TrueNorth neuron and synaptic weights integration (illustration adapted from [8]).

To reach high precision with very few bits of representation, it is now generally accepted that quantization-aware training [12] is key. Ternary [13] or even binary [14] values for both weights and activations, when used through hidden layers of convolutional feed-forward networks, can give very good results. These approaches are so much area and power efficient at performing the MAC operations that it is not needed to add control-oriented optimizations (e.g. skipping multiplications by zero).

The learning phase however still requires massive amounts of high-precision computations, which embedded devices generally can't afford. Gupta *et al.* [15], however, have shown that training with reasonably good accuracy is possible with 16-bit fixed-point operations, which helps to bring learning within reach of embedded devices.

III. Towards On-Chip Memory Only

Machine learning activities are notoriously heavy computing-intensive but also memory-intensive. Large modern inference networks involve tens to hundreds of millions of parameters and entails using separate processor and memory chips. But access to external memory consumes orders of magnitude more than reading from on-chip memory or doing calculations and this communication channel is often a bandwidth bottleneck.

The memory bottleneck issue is addressed in [16], where all neuron weights are stored in off-chip DDR memory. They perform design space exploration with caching, scheduling and data arrangement techniques to achieve the highest throughput possible.

To overcome both power and bandwidth limitations, there is a growing trend to use on-chip memory as much as possible, however at a high cost in area. In [17] only the convolution parameters are stored on-chip while all parameters for FC layers are kept off-chip. Convolution parameters are those accessed most often, so this trade-off brings highest power savings.

Han *et al.* [18] have proposed to exploiting the weight sparsity to compress the memory footprint and reduce the power due to DRAM access. They report an average 5 W power consumption for the computation of a fully connected layer, thanks to the fact that the weights fit in local SRAM

instead of DRAM (5 pJ/access vs. 640 pJ/access in their 45 nm technology), leading to 3 to 5 GOP/s/W on a Tegra K1 GPU. Using a limited number of different weight values to re-code and compress the memory of weights has also been proposed [19]. They achieve 177 GOP/s/W on a dedicated 45 nm chip that includes 64 processing elements and that is capable of running the first fully connected layer of VGG16.

Quantization is also very effective to reduce the memory footprint and help fit as many parameters on-chip as possible. Sung *et al.* [12] studied the trade-off between memory size and accuracy for various quantization levels ranging from 2 to 8 bits. The 2-bit case achieves Pareto-optimal results on a broad range of small networks, but 3-bit and 4-bit are slightly better to reach higher accuracy levels. Non-linear quantization also brings high memory saving while still providing a wide range of weight values, which is good for accuracy but at the cost of dedicated hardware support [17].

SqueezeNet [20] is an example of an approach which combines the exploitation of weight sparsity, weight sharing, and deep compression. They report a $510\times$ model reduction for AlexNet, leading to less than half a megabyte of storage for the same accuracy, but they do not give performance figures.

Ternary and binary quantizations enable to target on-chip only solutions, which allows to reach unprecedented memory bandwidth measured in tens of Tbit/s. There is however a downside: the MAC operators are so small that common memory compression techniques are difficult, if not impossible to implement efficiently. The resulting memory usage may even be higher than in other control-oriented approaches that focus on compressing high sparsity parameters.

A right choice of the type of layers in a network can also have a dramatic impact on memory requirements. For instance the most costly FC layers can be replaced by convolution layer [21] or average pooling [22], with only limited or no impact on accuracy.

IV. System Integration And Trade-Offs

To bring AI capabilities closer to the embedded world, dedicated solutions have been proposed.

TrueNorth [8] is a dedicated integrated chip for spiking neural networks. It features a consumption of only 65 mW for performance of 46 GSOP/s/W (synaptic operation per second and per Watt). But the spiking technology brings downsides, such as massive chip size (and corresponding price) and limited speed.

Recent works in binary and ternary neural networks on FPGA exhibit higher speed and often better higher accuracy. These approaches can also reach much higher power efficiency, in TOP/s/W, but only when using the hardware at its top speed to mask the high static power.

Orlando [17] is an accelerator for integer-based neural networks. It features on-chip memory, 16-bit multipliers, dedicated support for weight compression, non-linear quantization, convolution accelerators and embedded camera interfaces. This high level of integration enables to reach up to 2.9 TOP/s/W at 41 mW while running AlexNet on dataset ImageNet.

YodaNN [23] is a VLSI implementation of a CNN accelerator which make use of binary weights stored in standard latches. Thanks to that, it reaches a peak performance of more than 60 TOP/s/W. It handles three convolution kernel sizes ($3\times 3, 5\times 5, 7\times 7$) to support several CNN architectures, but cannot handle some, like Inception or SqueezeNet, that make use of kernels of size 1×1, 3×1, and 1×3.

The size of the window is directly linked to per-layer memory usage, however it is unknown how this impacts the number and size of the layers required to reach a given accuracy. The number of published fully integrated ASIC solutions is also too low to interrelate this trade-off with the performance of low-level hardware acceleration designs.

The choice of computing technology is of utmost importance for demanding inference tasks such as audio or video. However even small-scale networks can fit the needs in emerging wearables and IoT trends. The inference part is then just one element of a much larger system, which includes other power-consuming elements such as power supplies, sensors and wireless communication. Magno *et al.* present in [24] an emotion detection system that features a 100-neuron inference network implemented in software on a standard low-power microcontroller. So flexible and commodity hardware still has a place.

V. Emerging Processing-In-Memory Approaches

In addition to the approaches presented above, which reduce energy consumption by minimizing the number of MAC operations performed in DNN training and inference, or eliminate those operations by limiting the weights and activations to binary or ternary values, there are more disruptive approaches that focus on the reduction of power consumption using emerging non-volatile memories (NVM).

A. MAC Operations using Resistive RAM

Several approaches have been proposed to accelerate neural computing by means of NVM arrays. These approaches are

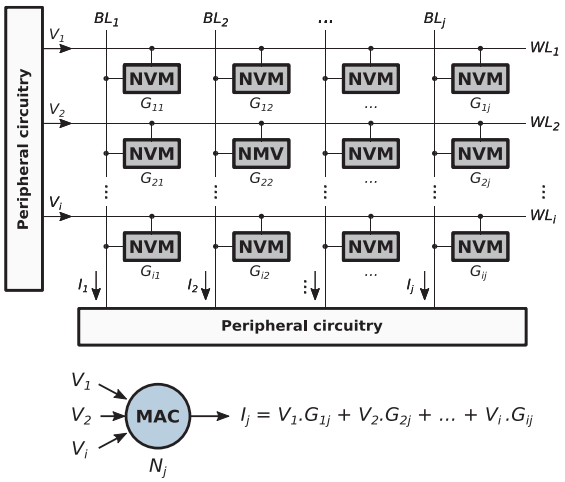

Fig. 5. Array of non-volatile memory devices.

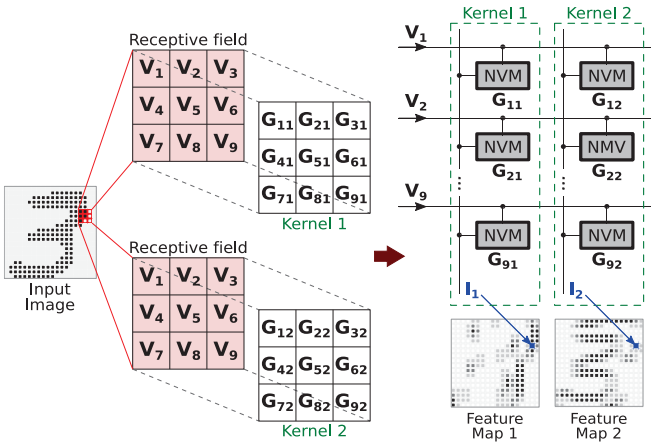

Fig. 6. Convolution into cross-point array (adapted from [31]).

different from the traditional Von Neumann ones, in which the synaptic weights are stored in RAM and transferred to the neural processing units repeatedly during computing.

The principle is to perform neural algorithms by executing analog computations directly at data location. The MAC operations can be realized and parallelized using Ohm's and Kirchhorff's laws, as advocated in Mead's pioneering work [25]. As shown Fig. 5, for an array of NVM elements, the MAC operation required by a neuron N_j is done by the addition of currents flowing through the column (bitline BL_j) associated to N_j. In this case, the analog input voltages V_i (presented on the wordlines WL_i) represent the input stimuli of a neuron N_j, and the conductances G_{ij} of the NVM elements represent the synaptic weights associated to each connection.

The design and deployment of NVM arrays in neural architectures has been addressed to reduce energy consumption, primarily considering an emerging NVM device called resistive random-access memory (RRAM). As opposed to phase change memory (PCM) cell arrays, for which access to the information uses additional transistors connected between the wordlines and the non-volatile storage elements [26], [27], NVM arrays based on RRAM do not require CMOS devices to access the storage elements.

Arrays of RRAM devices are also known as cross-point structures reducing the computation complexity from $O(n^2)$ to $O(1)$ in the case of MAC operations [28]. However, they could become energy-costly because they require more complex peripheral circuitry designs [29], which include ADC converters, subtraction units, and devices to avoid read disturbance. Xu *et al.* [30] show how the peripheral circuitry design controlling the access to a cross-point structure can also increase the chip area substantially, even though the memory cell size is five times smaller that the one containing a CMOS access device. High performance, low energy, and area efficiency are therefore not so easy to obtain with the cross-point structure.

By exploiting the possibility of performing MAC operations using resistive devices, Gao *et al.* [31] demonstrate how a CNN convolution kernel can be implemented in a 12x12 cross-point array. As shown Fig. 6, to determine the value of each input pixel on several feature maps, a receptive field (scanning the input image pixel by pixel) should be convoluted with different convolution kernels. The sum of products performed between the receptive field and each kernel will correspond to the pixel's value on each feature map. To implement these operations in parallel, receptive field is transformed into voltages applied on the cross-point array's rows, and each kernel into a column of resistive devices. The result of each convolution operation is determined by sensing the current on each cross-point array's column. In this work, convolution kernels are set in a memory array using an iterative programming protocol that adjusts the conductance of the resistive devices through the application of a sequence of pulses of incremental width and amplitude [32]. Experimental demonstration of kernel convolution operation was performed by classifying only two MNIST handwritten digits, but accuracy and energy estimations are not reported in this work.

Chi *et al.* [33] also use 256x256 RRAM crossbar arrays to perform neural computation directly in memory. Beyond demonstrating how the main neural network operations can be implemented using cross-point structures, they propose a processing-in-memory (PIM) architecture, which includes several functional blocks required to access the structure and control the analog computation. In the case of MAC operations required in fully connected layers of a multi-layer perceptron (MLP) approach, their architecture includes a peripheral circuitry composed of a driver block, which handles and connects multi-level voltage sources to the wordlines of the crossbar array. The voltage sources are controlled to send the voltage required as input to each wordline, either to access the memory array or to compute. The matrices of synaptic weights are separated in two different crossbar structures, the first storing the positive weights and the second the negative ones. Because of that, another analog block is located in the peripheral to compute the difference of the two MAC operations, for all operations. In addition, other functional blocks are proposed to support the activation, convolution and max-pooling functions. Experiments demonstrated an important reduction of energy consumption ($875\times$) by using RRAMs for neural network computation compared to software solutions. This result was obtained evaluating different neural network designs: CNN and MLP on the MNIST database, and VGG-D on ImageNet.

The MAC operations performed on resistive memory arrays storing analog values of non-uniform conductance can induce errors in convolution due to the variability of the resistive devices. As an alternative, Ni *et al.* [34] propose a bitwise CNN implementation operating with binary weights and activations. The proposed CNN model, obtained by training with binary constraints [35], performs convolution by bitwise dot-product between binary filter weights and binary feature maps (represented by values -1 and $+1$). Results of convolutions are processed by normalization and max-pooling intermediate layers, and finally binary activations are generated. The

implementation is based on a digital RRAM crossbar. Besides, unlike traditional analog-digital converters, only buffers and comparators are used in crossbar array periphery. To achieve convolution operations at a reduced energy cost, convolution is performed by an unsigned bitwise XNOR operation [36], where weights and inputs are converted to values 0 and 1. Since the signed bitwise convolution is avoided, two different RRAM crossbars are not required to represent positive and negative weights. Simulations results, obtained using the MNIST and CIFAR-10 benchmarks show accuracy of 98.3% and 91.4%. This bitwise CNN implementation achieves 126 TOP/s/W, which is more energy efficient than the best conventional approaches presented in [14] and [13], achieving 694 GOP/s/W and 1.23 TOP/s/W, respectively. Similar works, addressing the implementation of binary neural networks based on RRAM structures, are presented in [37], [38]. The first one achieves an accuracy 96.5% using a MLP of three layers (400, 200 and 10 neurons), but energy saving are not discussed. In this case, classification is performed on MNIST database by using a 512x1024 array of 16 Mb RRAM. The second one, which uses AlexNet model on ImageNet dataset, demonstrates 58.2% energy-saving by using 128x128 crossbar arrays. Overall, few experiments have been performed to demonstrate the use of NVM crossbar arrays at large scale and compare the energy gains obtained yet.

B. Artificial Synapse using Phase Change Memories

NVM arrays are also growing in importance throughout last years to model neuromorphic systems directly inspired by biology. In addition to being used for modeling synapses between neurons from an electronic perspective, they allow online learning by means of algorithms which fire pulses at the NVM elements to update their conductance [39]. Conductance in this case is related to the strength of the synaptic connection established between two neurons. Based on the input and output potentials (spikes) generated by the interconnected neurons when they are fired, and also on the timing at which these potentials are generated, the synaptic connections can be potentiated or depressed.

As demonstrated in [40], [41], [42], many efforts have been made to implement programming algorithms simulating the dynamic behavior of biological neurons. These implementations are usually related to the spiking neural network (SNN) architectures. Unlike DNN, the neurons are activated only when the accumulated electrical charges reach a predefined threshold, making of them energy-efficient models.

From a hardware perspective, PCM devices are one of the most popular candidates used to model electronic synapses, due to their ability to be gradually programmed in different states involving different conductance values.

PCM are also considered prominent candidates to be used in embedded systems due to their scalability to nanometric dimensions and their compatibility and integration with CMOS components. Specially, scalability of PCM arrays is discussed by several authors because it is not only limited by the scaling of the PCM device dimensions, it depends in part of reduction

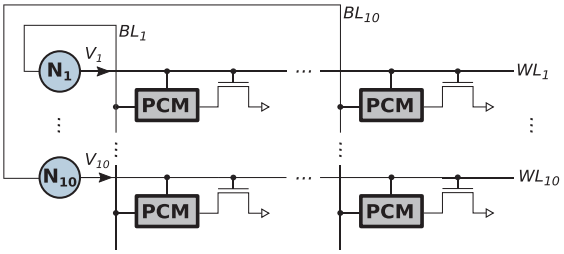

Fig. 7. Array of PCM devices interconnecting 10 neurons (adapted from [46]).

of diameters of electrodes, the reduction of the programming current and the reduction of the volume and composition of PCM materials [43]. Important analysis about the recent evolution of PCM in terms of retention, speed, power and data density are presented in [44], [45].

Pattern recognition in hardware using a 10x10 PCM crossbar array, with 10 interconnected neurons and then 100 synaptic devices is demonstrated in [46]. As shown Fig. 7, the input terminal of each neuron N_i is connected to the bitline BL_i and the output to the wordline WL_i. Besides, each synaptic element is composed of a PCM device accessed through a transistor. Learning in this case is performed using a Hebbian plasticity rule, where synapse is potentiated if interconnected neurons fire in the same time windows, or depressed otherwise.

An implementation demonstrating the large-scale use of PCM as synaptic elements is presented in [47]. The implemented model is a three layer perceptron including 916 neurons and 164885 synaptic connections. It includes a 500x661 crossbar structure using two different PCM devices to represent the weight associated to the synaptic connection between two neurons. Although 82% accuracy is demonstrated using the MNIST database and a backpropagation algorithm, these values are not compared with traditional networks based on conventional memories. Regarding power consumption, authors demonstrated that PCM-based on-chip machine learning can offer low-power (at least 120×) than GPU-based approaches [48].

VI. Summary

As this brief state of the art shows, the research for highly-optimized embedded ANN has lead to a great deal of scientific excitement especially these last two years. The most convincing results are obtained using conventional technologies in digital circuit design with hardware-aware training aiming at most area and power efficient solutions. More disruptive ideas aims at performing the computation right at the memory point in non-volatile memories. Overall, new challenges are ahead to bring still better AI capabilities in a growing number of embedded applications with high demands in terms of cost, power and versatility.

References

[1] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[2] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[4] O. Temam, "The rebirth of neural networks," Keynote speach at the International Symposium on Computer Architecture, 2010.

[5] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *ArXiv e-prints*, Sep. 2014.

[6] C. Szegedy, S. Ioffe *et al.*, "Inception-v4, inception-resnet and the impact of residual connections on learning," *CoRR*, vol. abs/1602.07261, 2016. [Online]. Available: http://arxiv.org/abs/1602.07261

[7] A. Pedram, S. Richardson *et al.*, "Dark memory and accelerator-rich system optimization in the dark silicon era," *IEEE Design & Test*, vol. 34, no. 2, pp. 39–50, 2017.

[8] F. Akopyan, J. Sawada *et al.*, "Truenorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 10, pp. 1537–1557, Oct. 2015.

[9] C. D. Schuman, T. E. Potok *et al.*, "A survey of neuromorphic computing and neural networks in hardware," *arXiv preprint arXiv:1705.06963*, 2017.

[10] M. S. Razlighi, M. Imani *et al.*, "LookNN: Neural network with no multiplication," in *Proceedings of the 2017 IEEE/ACM Conference on Design, Automation and Test in Europe*, 2017, pp. 1775–1780.

[11] G. Venkatesh, E. Nurvitadhi, and D. Marr, "Accelerating deep convolutional networks using low-precision and sparsity," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, March 2017, pp. 2861–2865.

[12] W. Sung, S. Shin, and K. Hwang, "Resiliency of deep neural networks under quantization," *CoRR*, vol. abs/1511.06488, 2015. [Online]. Available: http://arxiv.org/abs/1511.06488

[13] A. Prost-Boucle, A. Bourge *et al.*, "Scalable high-performance architecture for convolutional ternary neural networks on FPGA," in *27th International Conference on Field Programmable Logic and Applications*, 2017.

[14] Y. Umuroglu, N. J. Fraser *et al.*, "FINN: A framework for fast, scalable binarized neural network inference," in *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, Feb. 2017, pp. 65–74.

[15] S. Gupta, A. Agrawal *et al.*, "Deep learning with limited numerical precision," *CoRR*, vol. abs/1502.02551, 2015. [Online]. Available: http://arxiv.org/abs/1502.02551

[16] Z. Liu, Y. Dou *et al.*, "Throughput-optimized fpga accelerator for deep convolutional neural networks," *ACM Transactions on Reconfigurable Technology and Systems*, vol. 10, no. 3, pp. 17:1–17:23, Jul. 2017.

[17] G. Desoli, N. Chawla *et al.*, "14.1 a 2.9 tops/w deep convolutional neural network soc in fd-soi 28nm for intelligent embedded systems," in *IEEE International Solid-State Circuits Conference*. IEEE, 2017, pp. 238–239.

[18] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding," in *4th International Conference on Learning Representations*, 2016.

[19] S. Han, X. Liu *et al.*, "EIE: Efficient inference engine on compressed deep neural network," in *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture*, 2016.

[20] F. N. Iandola, S. Han *et al.*, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <0.5mb model size," *arXiv preprint arXiv:1602.07360*, 2016.

[21] J. T. Springenberg, A. Dosovitskiy *et al.*, "Striving for simplicity: The all convolutional net," *CoRR*, vol. abs/1412.6806, 2014. [Online]. Available: http://arxiv.org/abs/1412.6806

[22] H. Nakahara, T. Fujii, and S. Sato, "A fully connected layer elimination for a binarized convolutional neural network on an FPGA," in *2017 27th International Conference on Field Programmable Logic and Applications (FPL)*, Sept 2017, pp. 1–4.

[23] R. Andri, L. Cavigelli *et al.*, "Yodann: An ultra-low power convolutional neural network accelerator based on binary weights," in *IEEE Computer Society Annual Symposium on VLSI*, 2016, pp. 236–241.

[24] M. Magno, M. Pritz *et al.*, "DeepEmote: Towards multi-layer neural networks in a low power wearable multi-sensors bracelet," in *2017 7th IEEE International Workshop on Advances in Sensors and Interfaces (IWASI)*, June 2017, pp. 32–37.

[25] C. Mead, *Analog VLSI and Neural Systems*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1989.

[26] F. Bedeschi *et al.*, "A bipolar-selected phase change memory featuring multi-level cell storage," *IEEE Journal of Solid-State Circuits*, vol. 44, no. 1, pp. 217–227, 2009.

[27] N. Papandreu, H. Pozidis *et al.*, "Drift-tolerant multilevel phase-change memory," in *IEEE International Memory Workshop*, 2011.

[28] B. Li, P. Gu *et al.*, "RRAM-based analog approximate computing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 12, pp. 1905–1917, 2015.

[29] S. Kim, T. Gokmen *et al.*, "Analog CMOS-based resistive processing unit for deep neural network training," in *IEEE International Midwest Symposium on Circuits and Systems*, 2017.

[30] C. Xu, X. Dong *et al.*, "Design implications of memristor-based RRAM cross-point structures," in *Design, Automation & Test in Europe Conference & Exhibition*. IEEE, 2011, pp. 734–739.

[31] L. Gao, P.-Y. Chen, and S. Yu, "Demonstration of convolution kernel operation on resistive cross-point array," *IEEE Electron Device Letters*, vol. 37, no. 7, pp. 870–873, 2016.

[32] ——, "Weight tuning of resistive memories and convolution kernel operation on cross-point array for neuro-inspired computing," in *IEEE International Conference on Solid-State and Integrated Circuit Technology*. IEEE, 2016.

[33] P. Chi, S. Li *et al.*, "PRIME: A novel processing-in-memory architecture for neural network computation in ReRAM-based main memory," in *Proceedings of the 43rd International Symposium on Computer Architecture*, ser. ISCA '16. IEEE Press, 2016, pp. 27–39.

[34] L. Ni, Z. Liu *et al.*, "An energy-efficient and high-throughput bitwise CNN on sneak-path-free digital ReRAM crossbar," in *IEEE/ACM International Symposium on Low Power Electronics and Design*. IEEE, 2017.

[35] M. Courbariaux, Y. Bengio, and J.-P. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," in *Advances in Neural Information Processing Systems*, 2015, pp. 3123–3131.

[36] M. Rastegari, V. Ordonez *et al.*, "XNOR-NET: ImageNet classification using binary convolutional neural networks," in *European Conference on Computer Vision*. Springer, 2016, pp. 525–542.

[37] S. Yu, Z. Li *et al.*, "Binary neural network with 16 Mb RRAM macro chip for classification and online training," in *IEEE International Electron Devices Meeting*. IEEE, 2016.

[38] T. Tang, L. Xia *et al.*, "Binary convolutional neural network on RRAM," in *Asia and South Pacific Design Automation Conference*. IEEE, 2017.

[39] R. M. Shelby, G. Burr *et al.*, "Non-volatile memory as hardware synapse in neuromorphic computing: A first look at reliability issues," in *IEEE International Reliability Physics Symposium*, vol. 2015. IEEE, 2015.

[40] D. Querlioz, W. S. Zhao *et al.*, "Bioinspired networks with nanoscale memristive devices that combine the unsupervised and supervised learning approaches," in *IEEE/ACM International Symposium on Nanoscale Architectures*. ACM, 2012, pp. 203–210.

[41] O. Bichler, D. M. Suri *et al.*, "Visual pattern extraction using energy-efficient "2-PCM synapse" neuromorphic architecture," *IEEE Transactions on Electron Devices*, vol. 59, no. 8, pp. 2206–2214, 2012.

[42] D. Garbin, M. Suri *et al.*, "Probabilistic neuromorphic system using binary phase-change memory (PCM) synapses: Detailed power consumption analysis," in *IEEE Conference on Nanotechnology*. IEEE, 2013.

[43] R. Jeyasingh, J. Liang *et al.*, "Phase change memory: Scaling and applications," in *IEEE Custom Integrated Circuits Conference*. IEEE, 2012.

[44] G. W. Burr, M. J. Brightsky *et al.*, "Recent progress in phase-change memory technology," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 6, no. 2, pp. 146–162, 2016.

[45] A. Athmanathan, M. Stanisavljevic *et al.*, "Multilevel-cell phase-change memory: A viable technology," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 6, no. 1, pp. 87–100, 2016.

[46] S. Burc Eryilmaz, D. Kuzum *et al.*, "Experimental demonstration of array-level learning with phase change synaptic devices," in *IEEE International Electron Devices Meeting*. IEEE, 2013.

[47] G. W. Burr, R. Shelby *et al.*, "Experimental demonstration and tolerancing of a large-scale neural network (165 000 synapses) using phase-change memory as the synaptic weight element," *IEEE Transactions on Electron Devices*, vol. 62, no. 11, pp. 3498–3507, 2015.

[48] G. W. Burr, P. Narayanan *et al.*, "Large-scale neural networks implemented with non-volatile memory as the synaptic weight element: Comparative performance analysis (accuracy, speed, and power)," in *IEEE International Electron Devices Meeting*. IEEE, 2015.