Neuromorphic Computing and Engineering

TOPICAL REVIEW • OPEN ACCESS

The viability of analog-based accelerators for neuromorphic computing: a survey

NEUROMORPHIC
Computing and Engineering

OPEN ACCESS

RECEIVED
17 March 2021

REVISED
24 April 2021

ACCEPTED FOR PUBLICATION
17 May 2021

PUBLISHED
15 July 2021

TOPICAL REVIEW

The viability of analog-based accelerators for neuromorphic computing: a survey

Mirembe Musisi-Nkambwe*, Sahra Afshari, Hugh Barnaby, Michael Kozicki and Ivan Sanchez Esqueda

School of Electrical, Computer and Energy Engineering, Arizona State University, Tempe, AZ 85287-5706, USA
* Author to whom any correspondence should be addressed.

E-mail: mirembe@asu.edu, stafshar@asu.edu, hbarnaby@asu.edu, michael.kozicki@asu.edu and isesqueda@asu.edu

Keywords: analog computing, neuromorphic, synapse, memristor, non-volatile memory, machine learning, accelerator

Abstract

Focus in deep neural network hardware research for reducing latencies of memory fetches has steered in the direction of analog-based artificial neural networks (ANN). The promise of decreased latencies, increased computational parallelism, and higher storage densities with crossbar non-volatile memory (NVM) based in-memory-computing/processing-in-memory techniques is not without its caveats. This paper surveys this rich landscape and highlights the advantages and challenges of emerging NVMs as multi-level synaptic emulators in various neural network types and applications. Current and potential methods for reliably programming these devices in a crossbar matrix are discussed, as well as techniques for reliably integrating and propagating matrix products to emulate the well-known MAC-like operations throughout the neural network. This paper complements previous surveys, but most importantly uncovers further areas of ongoing research relating to the viability of analog-based ANN implementations based on state-of-the-art NVM technologies in the context of hardware accelerators. While many previous reviews of analog-based ANN focus on device characteristics, this review presents the perspective of crossbar arrays, peripheral circuitry and the required architectural and system considerations for an emerging memory crossbar neural network.

1. Introduction

Artificial intelligence is everywhere—using variants of deep neural networks (DNN) architectures for text prediction, object detection, speech and image recognition, to name a few. The computational tasks involved in conventional implementations of these neural networks require large data movements between memory and processing units. While there is continued development of dedicated hardware for these types of workloads, latency and power demands of this data traffic presents a well-known bottleneck and significant disadvantage especially for edge applications. Alternative architectures that perform matrix-vector-multiplication (MVM) in-memory using existing non-volatile memory (NVM) technologies may provide a solution to this bottleneck. The advantages and challenges of these analog NVM-based architectures are the main topic of this review paper.

As a complement to other surveys [1–6], the aim of this paper is to give a conceptual view of an analog-based artificial neural network (ANN) [1, 3, 6] from the perspective of the crossbar architecture and the individual NVM candidates for realizing the synaptic weights, as well as the means to propagate these weight products throughout the array in both directions. Also explored are the methods for *en masse* synaptic weight updates. The paper includes an overview of ANNs and DNNs for machine learning (ML) workloads, discusses ongoing research into analog-based ML hardware using existing NVM technologies and crossbar architectures and their limitations. A detailed review of the MVM macro explores candidate choices for synaptic weight storage to meet the requirements of different ANN applications. A cast of supporting peripheral circuitry follows—driving, sensing and data conversion architectures (ADC, DACs) with their architectural requirements and limitations. Concept analog-based accelerator architectures with their unique challenges are also evaluated and discussed.

The paper is organized as follows:

- Section 2 gives an overview of ANNs, current hardware challenges, and presents arguments for why explorations in analog based accelerators for ML are gaining traction.
- Section 3 describes current explorations in ML workload acceleration using analog hardware and the means in which the aforementioned features in section 2 are realized. It gives a snapshot of conventional memory storage and how the structure can be mapped into an ANN using analog devices in a conventional crossbar memory array.
- Section 4 describes the backbone of the analog hardware accelerator framework as being a conventional memory crossbar. This section describes how this is attractive for matrix vector multiplication (MVM) as well as relating the size limitations of the crossbar array to drivers, interconnect wire and source (synapse) resistances.
- Section 5 describes the requirements for ANN synaptic (weight storage) devices and presents candidates and qualifications.
- Section 6 presents the support circuitry required to drive, sense and modulate the synaptic devices and to perform computations based on the NVM synaptic weights. Circuits include data converters, drivers, sensors and where needed simple approximations of these circuits.
- Section 7 contains architectural and system considerations that address the unique analog-based ANN challenges—device variation and unresponsiveness, circuit non-idealities, precision control, effective multi-level signaling, signal regeneration and buffering, throughput, energy, area savings and latency. It also addresses challenges faced in current accelerators today and whether (and how) it affects an analog based NVM approach as well as what residual or new challenges remain for further research.

2. Background on deep neural networks

In simple terms, an ANN is a computing system formed by a collection of artificial neurons arranged in layers. Within each layer, each neuron takes inputs from all neurons in the previous layer, weighted by a scaling factor called the synaptic weight, constructing a weighted sum which for classification tasks is passed through a nonlinear activation function as shown in figure 1. This nonlinear (squashing) function, typically a softer expression of the sigmoid function, can be represented as a hyperbolic tangent, ReLU (or variations of) for improved classification accuracy. Or, as we will discuss later in the next section a simpler approximation [5]. The first and last neuron layers are called the input and output layers, and all intermediate layers are called hidden layers. In a single layer network the output neurons are simply a function of the weighted sum of the inputs. An ANN with multiple hidden layers indicates a deeper network, hence the term 'deep neural network' or DNN. The number of elements in a layer, especially the input layer is defined by the number of inputs or features, and can be further reduced to remove redundancy through various algorithms [7, 8] to break down into just the principal components that affect the intended output. The propagation of information from the input stimulus through the synapses from one layer to the next is referred to as forward propagation. Figure 1 illustrates forward propagation where the matrix of weights θ that map from one layer to another are multiplied by the inputs (from the previous layer), summed, and passed through the activation function to form an output matrix a_j^i. The weights, or synaptic values, represent the strength of the connection from one neuron to the next. The equations in figure 1 illustrate this mapping between layers and for simplicity only the mapping from the input to the first hidden layer and from the last hidden layer to the output layer are shown. In a multiclass classification task there would be several outputs represented by the hypothesis function h_θ in figure 1. The hypothesis function is a predictor that approximately maps the inputs to the outputs and is modeled or 'learned' from the test data provided in supervised learning.

The neural network cost function is $J_\theta = \mathrm{cost}(h_\theta(x), y)$ where we need to compute the θ (weights) that would minimize this cost function. Gradient descent is the general function to minimize the cost function in order to get the optimal synaptic weight matrix of θ for each layer. In the case of supervised learning the delta between the known and calculated output, known as the ground truth or 'labels' y_k, and calculated output, h_θ, are propagated backwards through the network. The back-propagation is actually doing two sweeps:

- Calculation and accumulation of error deltas for each layer in the equation multiplied by the partial derivative of the activation-function.
- And the second sweep is the synaptic weight update process.

This process is illustrated in figure 2.

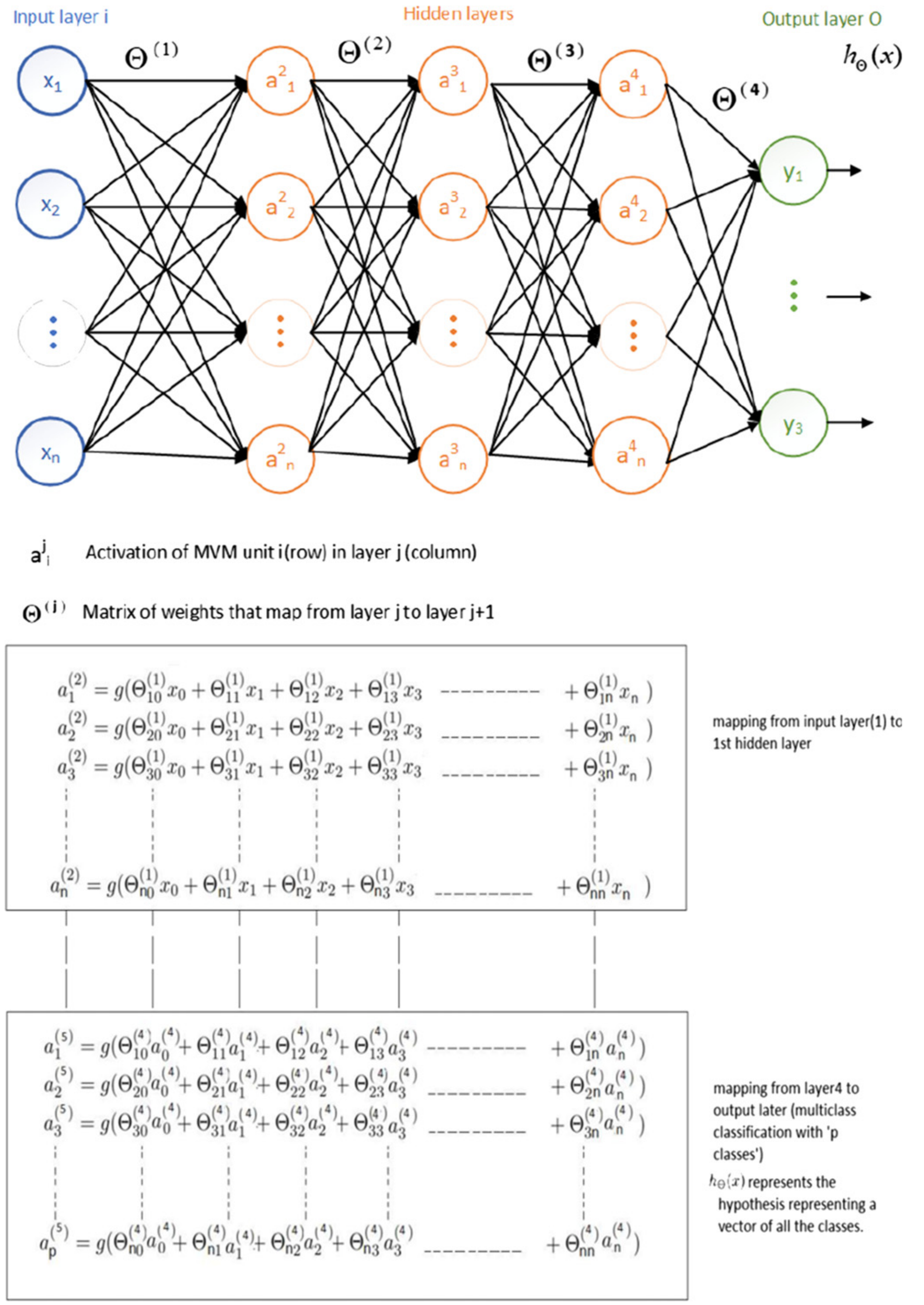
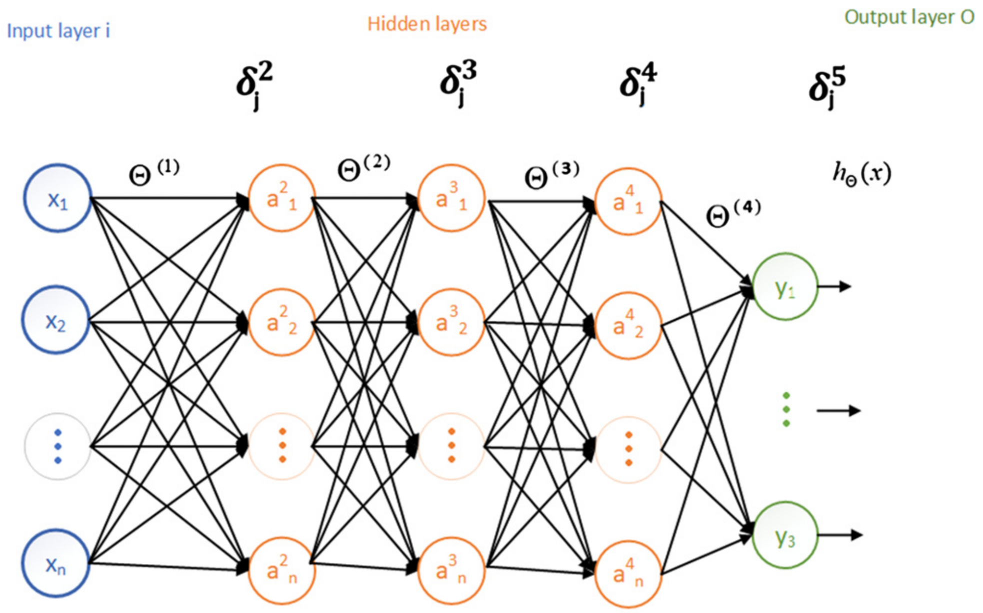

Input layer i
Hidden layers
Output layer O
$h_\Theta(x)$
$\Theta^{(1)}$
$\Theta^{(2)}$
$\Theta^{(3)}$
$\Theta^{(4)}$
x_1
x_2
x_n
a^2_1
a^2_2
a^2_n
a^3_1
a^3_2
a^3_n
a^4_1
a^4_2
a^4_n
y_1
y_3
a^j_i Activation of MVM unit i(row) in layer j (column)
$\Theta^{(j)}$ Matrix of weights that map from layer j to layer j+1
$a_1^{(2)} = g(\Theta_{10}^{(1)}x_0 + \Theta_{11}^{(1)}x_1 + \Theta_{12}^{(1)}x_2 + \Theta_{13}^{(1)}x_3$ --------- $+\Theta_{1n}^{(1)}x_n)$
$a_2^{(2)} = g(\Theta_{20}^{(1)}x_0 + \Theta_{21}^{(1)}x_1 + \Theta_{22}^{(1)}x_2 + \Theta_{23}^{(1)}x_3$ --------- $+\Theta_{2n}^{(1)}x_n)$
$a_3^{(2)} = g(\Theta_{30}^{(1)}x_0 + \Theta_{31}^{(1)}x_1 + \Theta_{32}^{(1)}x_2 + \Theta_{33}^{(1)}x_3$ --------- $+\Theta_{3n}^{(1)}x_n)$
$a_n^{(2)} = g(\Theta_{n0}^{(1)}x_0 + \Theta_{n1}^{(1)}x_1 + \Theta_{n2}^{(1)}x_2 + \Theta_{n3}^{(1)}x_3$ --------- $+\Theta_{nn}^{(1)}x_n)$
mapping from input layer(1) to 1st hidden layer
$a_1^{(5)} = g(\Theta_{10}^{(4)}a_0^{(4)} + \Theta_{11}^{(4)}a_1^{(4)} + \Theta_{12}^{(4)}a_2^{(4)} + \Theta_{13}^{(4)}a_3^{(4)}$ --------- $+\Theta_{1n}^{(4)}a_n^{(4)})$
$a_2^{(5)} = g(\Theta_{20}^{(4)}a_0^{(4)} + \Theta_{21}^{(4)}a_1^{(4)} + \Theta_{22}^{(4)}a_2^{(4)} + \Theta_{23}^{(4)}a_3^{(4)}$ --------- $+\Theta_{2n}^{(4)}a_n^{(4)})$
$a_3^{(5)} = g(\Theta_{30}^{(4)}a_0^{(4)} + \Theta_{31}^{(4)}a_1^{(4)} + \Theta_{32}^{(4)}a_2^{(4)} + \Theta_{33}^{(4)}a_3^{(4)}$ --------- $+\Theta_{3n}^{(4)}a_n^{(4)})$
$a_p^{(5)} = g(\Theta_{n0}^{(4)}a_0^{(4)} + \Theta_{n1}^{(4)}a_1^{(4)} + \Theta_{n2}^{(4)}a_2^{(4)} + \Theta_{n3}^{(4)}a_3^{(4)}$ --------- $+\Theta_{nn}^{(4)}a_n^{(4)})$
mapping from layer4 to output later (multiclass classification with 'p classes')
$h_\Theta(x)$ represents the hypothesis representing a vector of all the classes.

Figure 1. DNN learning network example illustrating forward propagation.

2.1. Types of networks

Figure 1 illustrates a fully connected (FC) DNN or multilayer perceptron (MLP) where each output activation corresponds to a weighted sum of all the inputs from its previous layer. The impractically large storage and computation requirements of these FC networks has prompted the exploration of sparsely connected network architectures. Convolutional neural networks (CNN) are an example of such sparsely connected architectures where weight sharing is used across an input feature map. The 'sharing' occurs whereby a filter of weights is convolving over a large input data matrix (see figure 3). The filters in figure 3 are simply pattern detectors and in DNN terminology, the filters correspond to the synaptic weights, while input and output feature maps correspond to input and output neurons. The abstraction levels of the input data (feature maps) are convolved with various filters in each layer, hence the various channels in various implementations [4]. As one gets deeper into the network these feature maps generate a higher level of abstraction of the input data. For example,



Weight update equation for layer 4: $\Delta\,\theta_j = \eta\ \delta_j^4 \ .*\ \text{sigmoid}(\theta_j^3\, a^3) = \eta\ \delta_j^4 \ .*\ a^4$

η learning rate

"error" of node j in layer l is calculated as such:

$$\delta_j^5 = a_j^5 - y_j = \text{sigmoid}(\theta_j^4\, a^4) - y_j$$

$$\delta_j^4 = (\theta_j^4)^T\, \delta_j^5 \ .*(\text{derivative\_sigmoid}(\theta_j^3\, a^3))$$

$$\delta_j^3 = (\theta_j^3)^T\, \delta_j^4 \ .*(\text{derivative\_sigmoid}(\theta_j^2\, a^2))$$

$$\delta_j^2 = (\theta_j^2)^T\, \delta_j^3 \ .*(\text{derivative\_sigmoid}(\theta_j^2\, a^2))$$

note that a, θ and δ are vectors

Figure 2. Back-propagation delta generation.

the low-level filters in the initial layers of CNNs used for image recognition may correspond to the image edges (e.g. horizontal and vertical) then, with deeper progression into the network these would correspond to more sophisticated shapes, and in the latter part to full objects. FC layers are found at the latter layers of the convolution network, usually the last one-to-three layers, as they are used for classifications hence also called ‘classifier layers’ illustrated in figure 3. Additional means to save storage memory in CNNs is using sub-sampling or ‘pooling’ to reduce feature map dimensions, illustrated in figure 3 solutions use an average, or max solution for the stride [4] in order to further reduce the input matrix to the next layer. In summary, there are three main layers in a CNN: convolutional layers, pooling (or sub-sampling) layers and a few full connected classifier layers at the end of the network.

Recurrent neural networks (RNN) are FC networks with large internal memory requirements to capture long term effects thus creating a computational bottleneck for today’s hardware accelerators. They require storing outputs from intermediate operations within the network to be used in processing of subsequent inputs, for example in natural language processing (NLP) algorithms. A feedback from the output to the input of the network allows for inhibiting or promoting parts of the input data based on history.

For event driven processing, spiking neural networks (SNN) are more favorable where information is spatio-temporal so active power becomes directly proportional to spiking activity, e.g. in event based vision sensors. In an SNN, an output neuron fires when the sum of its connections overcomes a threshold. An output

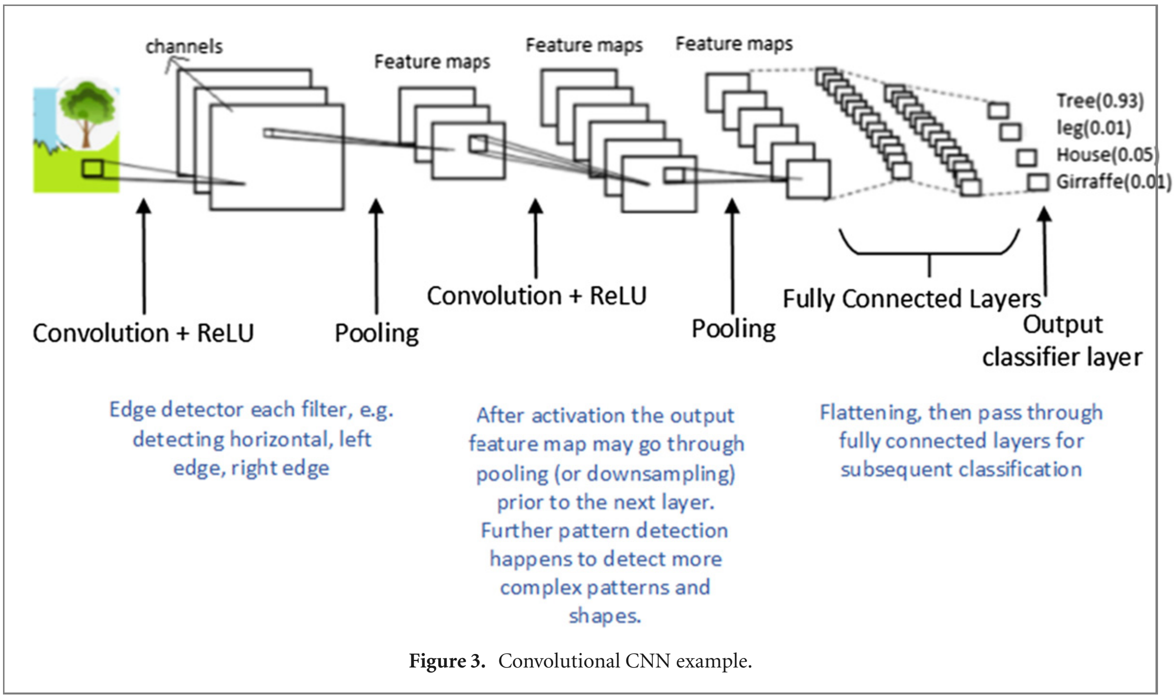

channels
Feature maps
Feature maps
Feature maps
Tree(0.93)
leg(0.01)
House(0.05)
Girraffe(0.01)
Convolution + ReLU
Pooling
Convolution + ReLU
Pooling
Fully Connected Layers
Output classifier layer
Edge detector each filter, e.g. detecting horizontal, left edge, right edge
After activation the output feature map may go through pooling (or downsampling) prior to the next layer. Further pattern detection happens to detect more complex patterns and shapes.
Flattening, then pass through fully connected layers for subsequent classification

Figure 3. Convolutional CNN example.

potential travels along connected synapses whose strengths could be inhibitory or excitatory after which the firing neuron's potential is reset. For forward inference systems one needs to factor in the firing frequencies and timing between the pre-and-postsynaptic spikes. In the case of training, to avoid the complexities of gradient descent, conventional DNNs are typically trained using back-propagation and subsequently the neurons are converted to spiking ones [9]. Local learning rules such as spike timing dependent plasticity (STDP) [10] are also used. SNNs can also be convolutional, SCNN, where pooling will have to be restricted to average pooling solutions rather than e.g. max pooling due to the spiking nature of the stimuli.

2.2. Popular models and data sets

Different architectures for ANN models have been studied and many are now featured as reference models for the benchmarking of inference and training AI hardware implementations [11–13]. The network architecture model is defined in terms of the number of layers, depth, layer shape (filter number and size, number of channels) and layer connectivity (e.g. FCNs vs RNNs, vs CNNs) and thus have different memory capacities and configurations requirements as seen in submitter benchmarking data in [11]. Various popular models exist (e.g. LeNet5 [14], AlexNet [15], VGG [16], GoogleNet [17], ResNet50 [18], Bert-99 [19]) and are compared and tabulated in [4].

Table 1 illustrates some of the datasets discussed in the various works discussed in this paper. Popular data sets discussed in this paper are those that are specifically used for analyzing novel analog-based ANN implementations and thus tend to be smaller and more rudimentary than the ones used for conventional/commercial purposes [20] as they are used for proof of concept. These include versions of MNIST, IMageNET, and CIFAR10/100 datasets.

2.3. Hardware

A brief historical timeline of neural networks is provided in table 2 to provide context to this paper. As indicated in the table, current trends are toward custom ASIC implementations to improve computational and power efficiencies and convergence rates of modern hardware accelerators. Accelerators are used for two main applications:

(a) Forward inference of pre-trained DNN,

(b) To accelerate the DNN training.

For each case, the hardware requirements are different and attract different applications [3].

Forward inference tends to be in a more power constrained envelope for use in edge, internet-of-things, and autonomous vehicle applications, as well as server room. These forward inference applications favor using hardware architectures with reduced latency over increased throughput (especially in edge computing). Training, which typically happens in the cloud [33, 34], relies on hardware designed for throughput (ops/sec) over latency, with usage of distributed multiple compute nodes optimizing the intercommunication between them.

Table 1. Sample popular data sets.

Dataset name	Dataset category and task	Description	Instances
TIMIT [21]	Speech recognition, classification	Collection of phonemes from eight major dialectsof American English.630 speakers each reading ten phonetically rich sentences	6300
Yale face database [22]	Facial recognition	Faces of 15 individuals in 11 different expressions	165
MNIST [23]	Handwriting and character recognition, classification	Database of handwritten digits	60 000
CIFAR-10/100 [24]	Object detection and recognition, image classification	Low resolution images of 10 (CIFAR-10) or 100 (CIFAR-100) object classes	60 000
KITTI [25]	Object detection and recognition	Images and scenes captured with cameras and laser scanners for autonomous vehicle usage and laser scanners for autonomous vehicle usage	>100 GB of data
ImageNet [, 26]	Object detection and scene recognition	Labeled object image database over 20 000 categories	1.4×10^7
Free music archive	Music, classification and recommendation	Audio from 100k songs with hierarchies from several genres, metadata, and user information	5665

Edge in-the-field training is gaining more traction [35–37], not only due to the latency of training in the cloud, but also due to privacy/security risk concerns, and to reduce reliance on connectivity.

Existing hardware used for implementation of neural networks includes CPUs, graphics processing units (GPU), and tensor calculation specific ASICs [30, 33, 36]. These are generally enhanced using special software drivers and stacks provided through various libraries [4, 38, 39]. GPUs accelerate ANN implementations using massive parallelism of processing cores optimized for computing applications. This is different from traditional CPU multi-core processors which are more generic. The handling of floating point operations in GPUs is also attractive for the implementation of neural networks as it enables larger and deeper networks with many neuron computations performed in parallel. While GPUs were created to accelerate graphic rendering, TPUs are AI accelerator ASICs specifically designed for tensor calculations, and developed to accelerate deep learning workloads.

To provide a means of benchmarking performance for ML workloads, a consortium called MLPerf [12, 13] specifies reference model architectures and data-sets to provide industry standards for measuring and comparing ML performance.

2.4. Current challenges

Key metrics and challenges for today's ML accelerators are latency, energy consumption, and throughput. Within inference applications, where latency is crucial especially for online applications, there are allowances for reduced precision in matrix calculations while still maintaining classification (prediction) accuracies. To address these challenges, data is encoded using smaller bit-widths with use of fixed-point versus floating point representations [3, 40] for synaptic and activation function precision. Pruning the network removes neurons that are not important using sparse matrix methods, or as described in [4], studying weight saliency and setting the less significant weights to zero or just skipping over these weights entirely during computations. The usual trend to gain sparsity is to increase the number of convolutional layers and decrease the number of fully-connected (FC) layers, which additionally decreases memory fetches and memory bandwidth. This is not a viable option for applications that require FC networks (e.g. RNNs). So, with the increased latency of memory fetches (with growing depth in neural networks) other means of increasing memory bandwidth need to be investigated.

In training applications the aforementioned techniques must be done with care as higher precision requirements are needed for gradient descent and other optimization approaches. One approach is to replace stochastic gradient descent (SGD) with batch or mini-batch gradient descent where the loss is calculated from multiple sets of data before doing a weight update to stabilize and speed up the process [4]. Sparsity can also be gained from reducing the complexity of the sigmoid function to a ReLU function which gets negative values to zero. Another method is feature extraction down to principal feature components, or other means of compression [40].

Table 2. A timeline history of neural networks.

DNN timeline	
1940s	Neural networks (NN) proposed, e.g. Hebbian learning/Hebbian networks and computational models for NN
1960s	DNN proposed, GMDH (group method of data handling— earliest DNN of feed forward MLP) [26]
1979	CNN predecessor neocognitron introduced
1980s	NN specific application of back propagation
1989	Neural networks (CNN) for recognizing handwritten digits (LeNet)
1990s	CNN with backpropagation LeCun's refining of LeNet. [23] dataset introduced. MNIST Max pooling CNN (Cresceptron) hardware for shallow neural nets (Intel ETANN [27] at IJCNN conference)
2000s	GPU implementations offer 4× CPU speeds, K Chellapilla. ICDAR 2009 pattern recognition contest
2011	Breakthrough DNN-based TIMIT [21] speech recognition (Microsoft), ImageNet [26] contest, improvements in visual object detection within a large image-ICPR 2012 contest
2013	MICCAI 2013 grand challenge on mitosis detection and also recognition of distorted text in reCAPTCHA puzzles
2012	AlexNet, eight layers— five convolutional and three fully-connected, f irst to use Rectified linear units (ReLUs) for activation functions
2014	GoogleNet, 22 layers deep network w inner of ILSVRC 2014 [17]. NVM based accelerator exploration growth [28, 29]
2015	Processor limitations cause a growth in DNN accelerator research optimized for neural network applications specific ASICs tensor processing unit [30] (Google), (Neuflow [31], DianNao [32]). Continued exploration in non-conventional approaches
2016+	Edge specific AI/computing, internet of conscious things

These challenges ultimately mean changes need to be made to the hardware architecture to ensure advances in improved throughput, latency and energy consumption are at lockstep with the inevitable complexities of growing datasets [41].

2.5. Near data processing and the promise of in memory processing

As suggested in the previous section, existing solutions favor *memory light* approaches with reduced precision (where possible), and rely on pruning, data compression, and structured sparsing techniques. Current accelerators [4] integrate different levels of local memory along processing element (PE) routes as shown in figure 4. In these 'near memory' implementations, data can be routed between ALU, register file, and PEs for cheaper memory accesses. In examples like [42], where local memories are interspersed through the tensor processing cores and larger high-bandwidth memories around the periphery, there is limited capacity for these low cost memories, so the trend is to exploit data reuse to reduce memory fetches by using convolutional architectures where relevant. However, with growing demand for higher throughput, larger data sets, and need for reduced latencies, these types of implementations will no longer be enough and face a familiar memory bottleneck. If computation can be done within the storage unit, significant improvements will be achieved for latency, throughput, and energy consumption (i.e., the three main challenges for today's accelerators). This approach, known as in-memory computing, and its own novel challenges (such as data regeneration, data conversion, device and circuit variability, etc) are discussed in the subsequent sections.

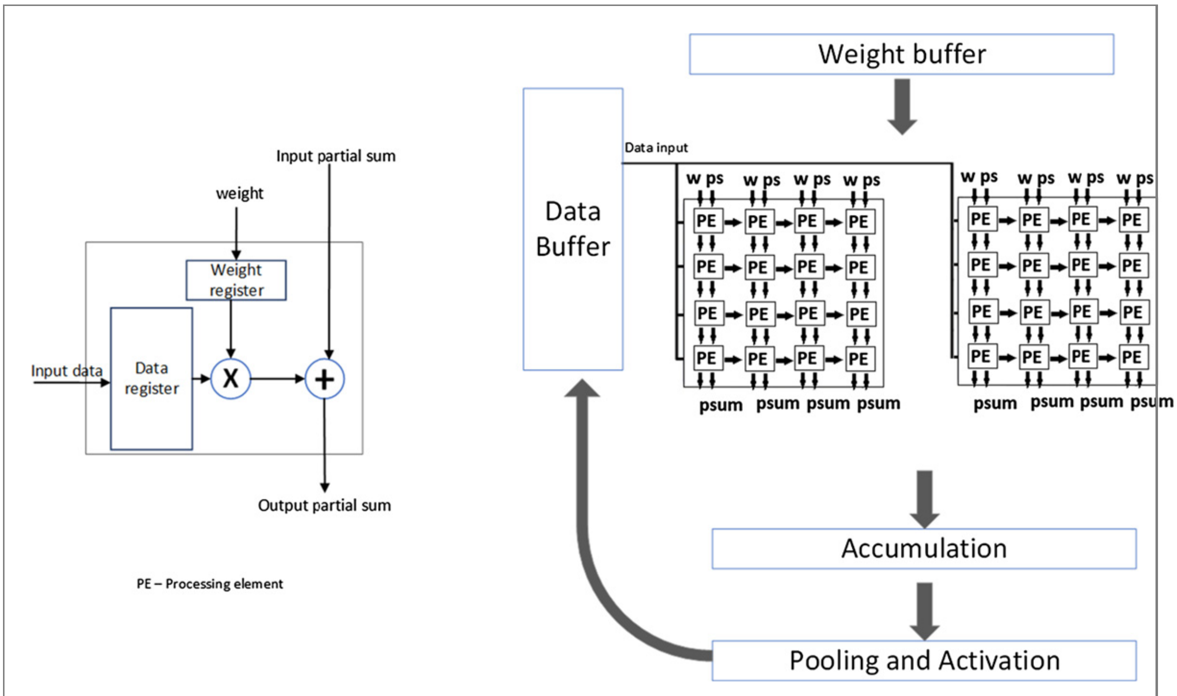


Figure 4. Basic diagram and network inside a hardware accelerator, showing the internals of the PE doing an MAC operation on the left. Then its instantiation within the matrix multiply unit internally can be presented as a convolutional or FC or both.

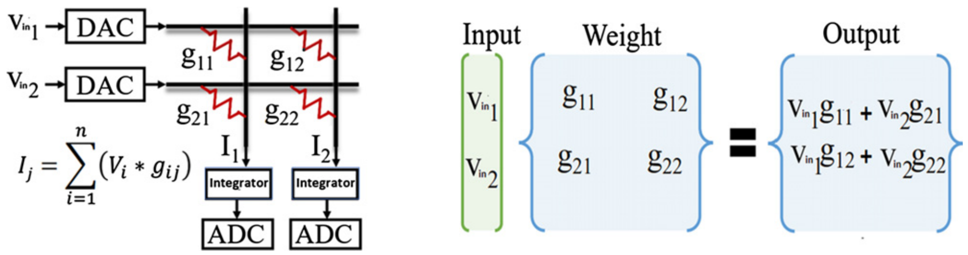


Figure 5. Matrix vector macro unit and its mapping to a matrix.

3. Analog hardware for in memory processing of ML work loads

Constant fetches to memory to access weights and partial sums when performing matrix calculations introduce latencies due to the high data movement. While attempts have been made to mitigate this bottleneck with specialized AI accelerators [30, 42] based on near-memory computing, growing data-sets and computational requirements have forced traction for the development of in-memory computing systems [43]. Discussions in section 2 mentioned even with NN memory light solutions such as convolutional networks (CNN) some applications need FC layers—RNN (LSTM, GRU).

The diagram in figure 5 shows a concept diagram of an analog based DNN with resistor processing elements (RPE) driven and sensed by peripheral circuitry in both directions.

In essence the 2D matrix calculation from equation in figure 1 is mapped into a physical RPE array where the conductance element (RPE) at the crosspoints represents the synaptic weight between the row and column. This configuration is typically called a crossbar array or simply a crossbar. The weight is encoded into the device conductance and in many cases it requires a multi-bit value for higher accuracy and resolution. Two floating point operations (multiply and accumulate) can be *condensed* into one parallel operation as shown in the diagram in figure 6. Moreover, these operations can be done in *parallel* for all columns in the crossbar array resulting in parallel multiplications of input vectors with the weights matrix (vector-matrix-multiplication, or VMM) implemented in one step. Thus, this in-memory analog implementation of VMM avoids moving weights from memory to separate processing units and enables large *parallelism* in the computations.

Several existing storage memory crossbar hardware have already been shown to model the above matrix operations and can be used to do matrix vector multiplications *in situ* [44–48]. These are based on various storage devices to implement the weights. Thus, in these crosspoint technologies, each memory cell at the

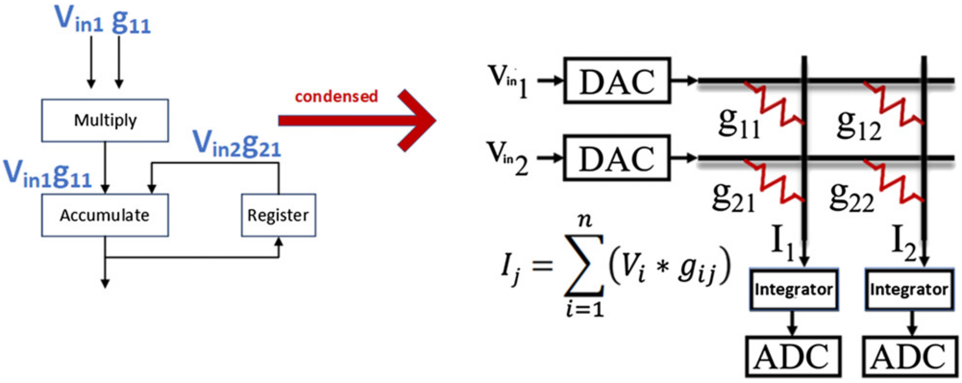


Figure 6. The condensation of two floating point operations—multiply and accumulate into a single parallel operation in the crossbar.

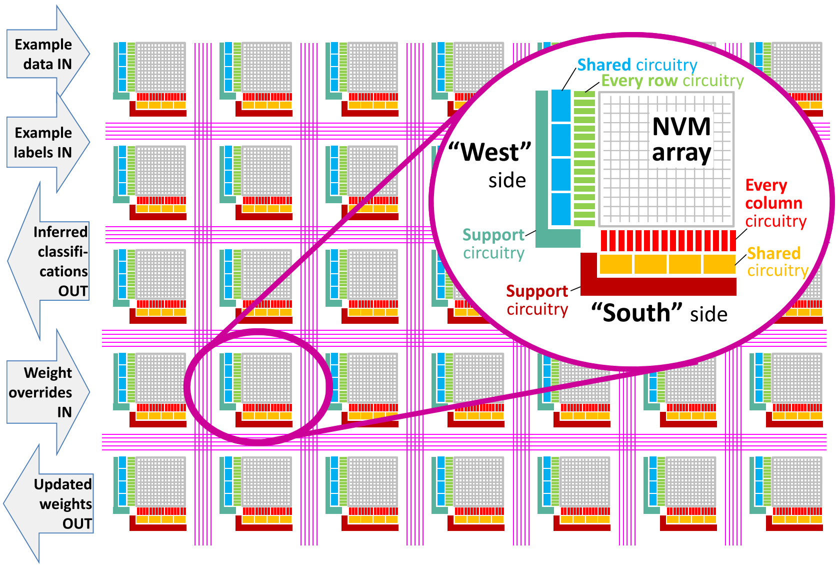


Figure 7. An array of DNN blocks interconnected and reconfigurable. © 2017 IEEE. Reprinted, with permission from author, from [5].

row–column intersection holds the weight of a synapse and can be manipulated based on the device characteristic to provide multiple states. These states typically correspond to device conductance state (e.g., in filamentary or charge-based resistive switching memory). Further illustrated in figure 7 from [5] is a generic architecture for DNN training using NVM based arrays where the architecture is split into array-blocks (large NVM array) that are interconnected by a flexible routing network. The routing fabric is to transfer input-data, weight updates from chip inputs into the device array and to carry updated chip information and inference classifications out. The flexibility is allowing for reconfigurability to multiple layers to control the depth of the neural network. The design grid connects input neurons on the west side of the array block to the output neurons on the south side each being fed by peripheral circuitry to drive and sense. Local storage is required for the activation excitation and error value during an *in situ* training application so that it can be used and compared later for weight updates. This will be further discussed in the architecture section 7.

3.1. Forward inference networks

Inference solutions begin with physical synaptic elements/devices being programmed with weights obtained from an *ex situ* training solution (typically done in software). The details and methods of mapping will be briefly discussed in section 7. One of the earliest methods for an inference accelerator was IBM's TrueNorth [28] where a large SNN was implemented using an SRAM crossbar array to perform forward propagation. The weights were trained offline and transferred onto the SRAM array that corresponded to 256 million synapses and 1 million neurons. Such attempts at CMOS-based synapses and neurons in neuromorphic systems [28, 49, 50] are not area efficient due to the large number of transistors needed for their implementation.

In analog-based implementations, the focus of this paper, a more area efficient solution is explored. There, SRAM cells are replaced with analog memory, not only to save area but also to extend beyond binary weights (i.e., dual-state representation of weights) [5, 45, 51] to allow more granularity and precision, as well as to enable in-memory neuromorphic computing architectures using crossbar configurations. The architecture in [45] takes this further to provide a solution on demand that can be dynamically reconfigured between accelerator and memory supporting MLPs (FC NN) and CNNs using resistive-RAM (ReRAM or RRAM). Once synaptic weights are written and verified to be mapped correctly, the inference phase will drive read signals from a DAC (non-disturbance signals) in order to read the current 'setting' of the synaptic weight element. For example, considering a ReRAM crossbar array, a driving voltage signal would be applied to the rows of the crossbar activating current flow through each resistive element. The sum of these currents are collected at the end of the crossbar column and integrated on a capacitor which can then be passed directly to an analog approximation of the activation function [51], (or converted into a digital signal for a more logic approach [45]) prior to driving the next hidden layer. The synaptic elements can be stimulated in different ways for a read operation depending on the type of element. Encoding from the DAC can be amplitude-modulated or time-modulated depending on the type of device—for example ReRAM (resistive RAM) [45, 52] or phase change memory (PCM) respectively [51]. Note that each column is driven by a combination of the various elements and subsequently drivers feeding these elements. So each column will have its own calculation, and the same goes for rows in the reverse direction for backpropagation. In propagating to the next hidden layer architectures can save energy through circuit sharing by time multiplexing the ADC and/or activation implementations. To realize the positive and negative weights device pairing can be used [53], since the physical storage mechanism typically corresponds to a positive value. For example, in ReRAM implementations [45] two crossbar arrays are used to store positive and negative weights respectively, and their difference is obtained using a subtraction unit prior to passing over to the activation unit. Similarly in [51] two PCM devices are used, one as positive (LTP) and one as negative (LTD) contributing opposite effects at the integrator during a read.

3.2. Back propagation

As discussed in section 2, supervised learning issued by back-propagation of error terms is used to adjust the weights. In an analog-based training solution this learning happens *in situ* as the crossbar element states are adjusted, so hardware friendly approaches are required to implement learning algorithms such as those based on gradient descent [54]. As described in section 2 the back propagation is triggered by a calculation of errors propagated throughout the network from one layer to another. The column drivers propagate the error values though the synaptic weight in order to do a 'forward propagation in the opposite direction' and in a resistive solution, the current is accumulated on the row capacitor [51]. The error values accumulated on this row capacitor represent the accumulated error for propagation to the next neuron. This value can be sent to an ADC and further processed digitally by combining with the derivative of the activation function or using a simple circuit approximation [5, 51] step function to connect to the preceding upstream layer to create the accumulated error value for that neuron. The classification accuracies can be improved by mitigating the vanishing gradient problem by creating a leaky derivative emulation through redefining the 'zero' level of this step function [51]. The diagram in figures 8 and 9 illustrates this process. Note that for backpropagation some sort of local storage is needed for the activation and calculated error for use in the weight update calculations.

3.2.1. Weight update

In [5, 51] MLP DNN the upstream neuron sends a signal based on its activation value and the downstream neuron sends a signal based on its back-prop error value, the overlap of these signals is used to program the synapse. The relative temporal difference between the two determines the magnitude and whether this will be a *potentiation* of the synapse or a *depression*. In [55] is a more detailed study on the concept pairing a synaptic LTP vs synaptic LTD using a two-PCM synapse (crystallization phase to allow for gradual conductance and avoid the abruptness of LTD in amorphous phase) so essentially a PCM–LTP device in parallel to a PCM–LTD device see figure 10. The method is referred to as a modified STDP update rule.

During the LTP time window the interaction of a write pulse with the feedback pulse 'potentiates' (increases) the conductivity of the LTP device. During the LTP phase the lone feedback pulse by itself will only increase the conductivity of the LTD device thus depressing the equivalent synapse. Accommodating the two phases means longer write times, but the split is required due to driver/sensor stability problems at endpoints of a particular synapse and is an open area for further research. A means to reduce this latency is to investigate devices that support shorter set-pulse times [52, 56]. The effective change in conductance is studied in [57] with 1000 pulses to a phase change element and explores the effective change in conductance based on initial conductance value and the extent of causality and anti-causality firings to mimic the relative time slots of row

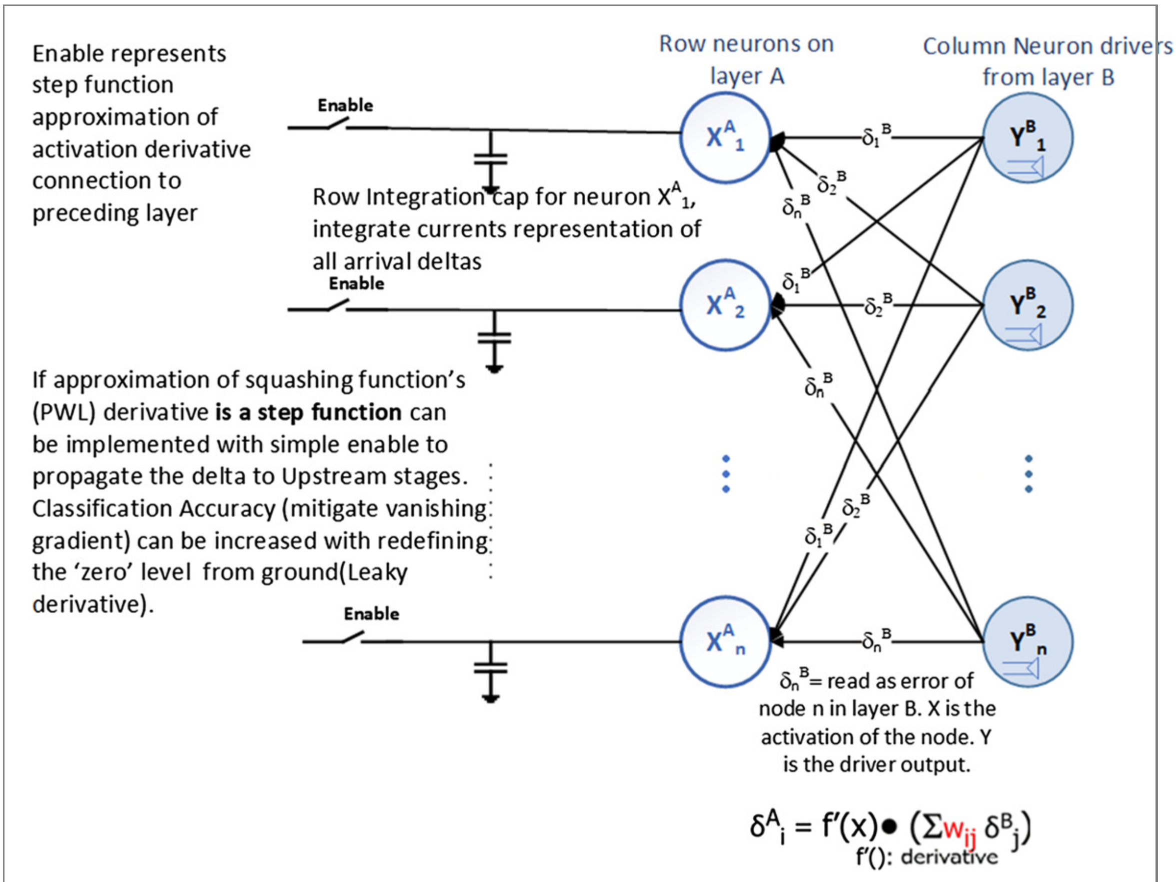

Enable represents step function approximation of activation derivative connection to preceding layer
Row neurons on layer A
Column Neuron drivers from layer B
Enable
Row Integration cap for neuron X^A_1, integrate currents representation of all arrival deltas
If approximation of squashing function's (PWL) derivative is a step function can be implemented with simple enable to propagate the delta to Upstream stages. Classification Accuracy (mitigate vanishing gradient) can be increased with redefining the 'zero' level from ground(Leaky derivative).
X^A_1
X^A_2
X^A_n
Y^B_1
Y^B_2
Y^B_n
δ_1^B
δ_2^B
δ_n^B
δ_n^B= read as error of node n in layer B. X is the activation of the node. Y is the driver output.
δ^A_i = f'(x)● (Σw_ij δ^B_j)
f'(): derivative

Figure 8. A diagrammatic concept view of a 2D crossbar concept during backpropagation error calculation.

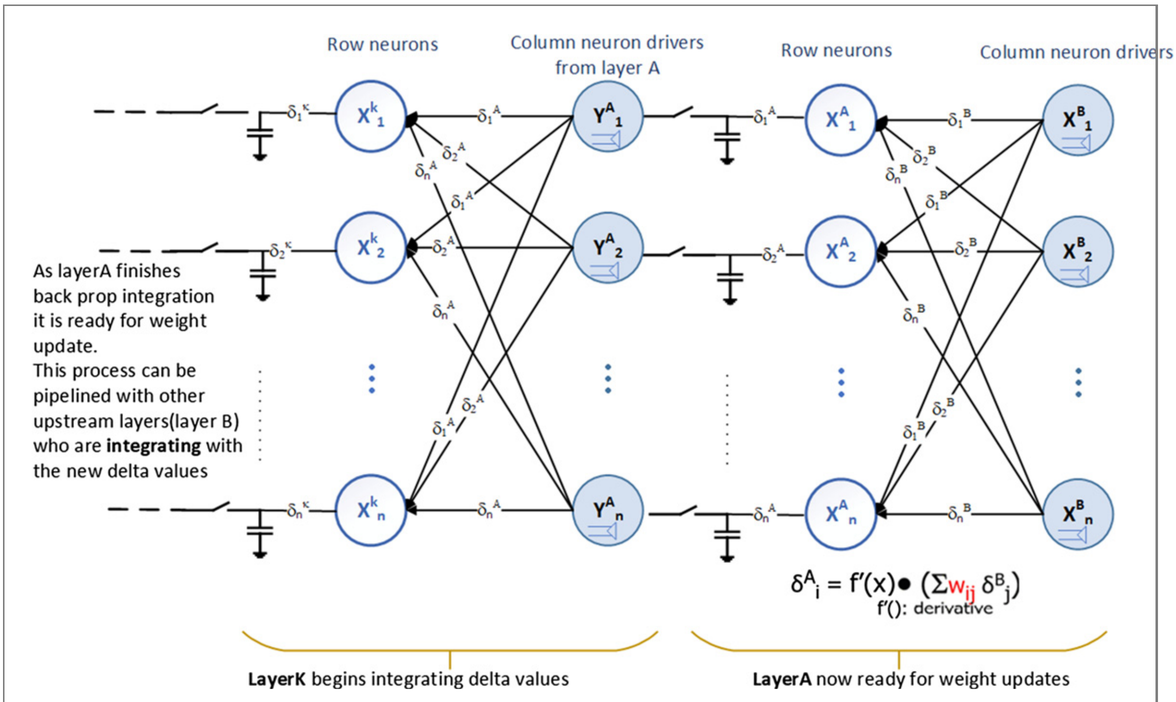

Row neurons
Column neuron drivers from layer A
Row neurons
Column neuron drivers
X^k_1
X^k_2
X^k_n
Y^A_1
Y^A_2
Y^A_n
X^A_1
X^A_2
X^A_n
X^B_1
X^B_2
X^B_n
δ_1^K
δ_2^K
δ_n^K
δ_1^A
δ_2^A
δ_n^A
δ_1^B
δ_2^B
δ_n^B
As layerA finishes back prop integration it is ready for weight update. This process can be pipelined with other upstream layers(layer B) who are integrating with the new delta values
δ^A_i = f'(x)● (Σw_ij δ^B_j)
f'(): derivative
LayerK begins integrating delta values
LayerA now ready for weight updates

Figure 9. A diagrammatic view of 2D crossbar illustrating a pipeline of backpropagation error calculation and weight update.

and column drivers. One could also use the selector device turn on [58] as an additional knob to control the amount of overlap. In [59] describes a similar means of doing a parallel write/updating where the encoding on either side of the ReRAM device is different—for the column driver as pulse amplitude modulation and row driver as pulse width modulation to effect the change in ReRAM conductance. The larger the amplitude the more the weight change as well as the duration of the pulse. [60] proposes a spike based read integrate fire

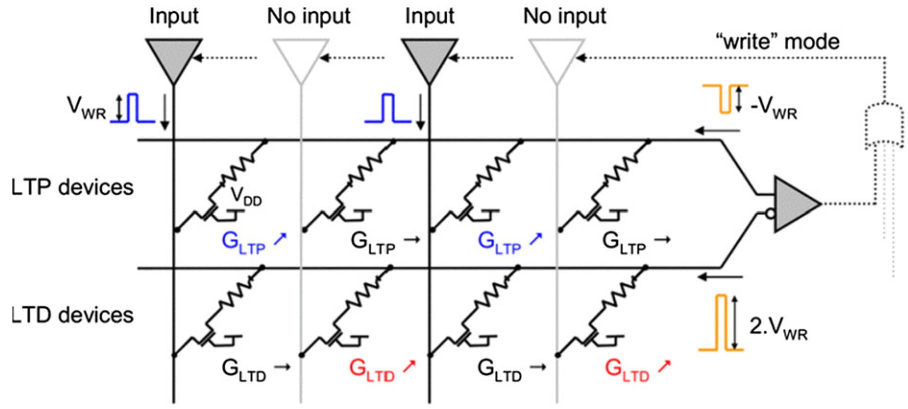


Figure 10. Modified STDP. © 2011 IEEE. Reprinted, with permission from author, from [55].

circuitry to represent the input current into digital spikes and in write mode, this spike train overlaps with a duty cycled feedback pulse to potentiate or depreciate the device (the polarity being controlled by the sign of the spike pulses). The [61] positive and negative weights are presented as a deviation from the 'zero weight state' described as the mid-point between the RON + ROFF state thus avoiding (where possible) the need for a device pairing as in solutions mentioned earlier. Memristive FET crossbar structure is investigated in [62] using pre and post synaptic spikes on drain and gate FET terminals. The modulation of the FET threshold voltage by changing the gate to drain voltage creates the STDP positive and negative STDP updates. The 'shape' of the spike can be used as an added hyperparameter knob to implement a faster or slower learning process as needed. The effectiveness of writes degrades with number of pulses where by the effective change in conductance decreases over pulses [63], this will be discussed in section 5 on how systems handle 'stuck-ats' and reset strategies for saturated paired conductances. While time consuming for online updates, offline training solutions (*ex situ* training) can reliably write using a read verify write to account for this prior to device mapping. A hybrid-training approach is discussed in [64] for a memristor-CNN where only the final FC layers are trained *in situ*.

Synaptic weight update pulsing and decisions on how many pulses, amplitude and shape are dependent on memory device type and technology. The next section will explore use of NVM 2D crosspoint technology for ML workload acceleration.

4. 2D crosspoint for ML acceleration

The support frame of the analog based accelerator architectures is the 2D crossbar array, the size of which is determined by its line resistance, synaptic resistance and driver resistance [65–67]. The arrays are driven on either end of the synapses by drivers and sensors fed by DACs and ADCs respectively to allow for bidirectional signaling. The crossbar size is dependent on the synaptic device's low resistance (R_{LRS}), its high resistance to low resistance ratio $R_{\text{HRS}}/R_{\text{LRS}}$ ratio, and the number of states that can be programmed and read reliably (which also affects latency and the required switching energy [63]).

4.1. Crossbar size limits

The ratio of the memristor resistance to the driver resistance also determines how large the crossbar can get as shown in figure 11 from [65]. A relation that predicts the maximum crossbar size relating the driver transistor to memristor resistance ratio, the write voltage to memristor threshold and the number of devices to be written in parallel, W, based on a large data set of 2000 points is expressed in [65]. In figure 11, increasing the R_{LRS} (synaptic 'on' resistance) to driver resistance ratio ($R_{\text{m}}/R_{\text{t}}$) allows for a larger crossbar due the reduced effective load resistance (interconnect resistance was not accounted for in this analysis) from greater number of memristors. But, an increased R_{LRS} reduces the synaptic resistance window and hence the number of realizable states/levels, limiting multi-state capability and classification accuracies [67]. The greater the number of devices to be written in parallel from a driver is also analyzed in the second figure, illustrating that there is a limited number of devices that can be supported above the write threshold.

4.1.1. Reducing effective crossbar line resistance

The minimum voltage required for both worst case memory cell (with its selector device if used) to switch is discussed in [66] and used as a minimum threshold for write voltage shown in figure 12. A figure of merit called

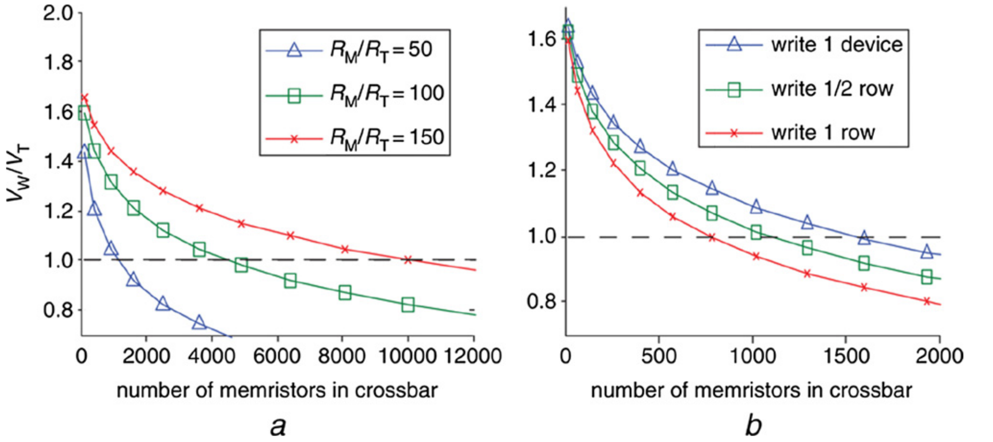


Figure 11. (a) Plots of write voltage to write threshold ($V_w/V_t = 1.9$) versus crossbar size for three memristor to driver ratios (R_m/R_T). (b) Given $V_w/V_t = 1.9$ how many devices can be written in parallel. © 2016 IEEE. Reprinted, with permission from author, from [65].

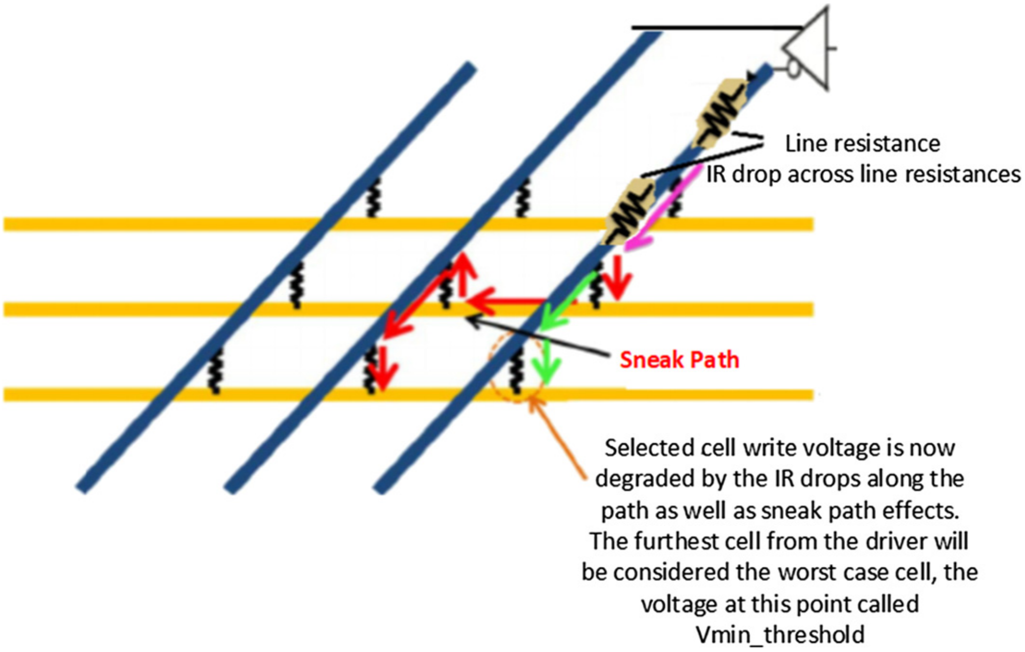

Line resistance
IR drop across line resistances
Sneak Path
Selected cell write voltage is now degraded by the IR drops along the path as well as sneak path effects. The furthest cell from the driver will be considered the worst case cell, the voltage at this point called Vmin_threshold

Figure 12. Effective voltage at the selected cell (memristor device + selector if any), which is degraded from the disturbance voltage at the driver due to sneak paths and metal line resistance (source and sink resistances on the driver and current sense circuitry also play their part in IR drop but can be handled in the peripheral circuit design).

the normalized write window as a means to evaluate the crossbar reliability is described as the write disturbance voltage (the maximum voltage drop on an unselected cell which happens to be closest to the driver) subtracted from the voltage at the selected cell (V_{cell}), divided by the disturbance voltage (V_{dis})

$$\text{Normalized write window} = \frac{V_{\text{cell}} - V_{\text{dis}}}{V_{\text{dis}}}.$$

This normalized write window described in [66] decreases with increasing array size due to the increase in interconnect resistance and hence reduced effective write voltage on the selected cell. This can be mitigated by using multiple drivers to reduce the effective interconnect length thus increasing the effective write voltages to the selected cell resulting in reduced write latencies and switching energy as illustrated in the case study from [66] in table 3. Using a dual row driver effectively changes the array from an $N \times N$ array to an $\frac{N}{2} - 1$ columns by $N - 1$ row. In the quad driver case this is further reduced in size to an $\frac{N}{2} - 1$ column by $\frac{N}{2} - 1$ rows shown in figure 13 from [66]. While this increases write power, the gains in switching speed are substantial. An increase in driver voltage in attempt to achieve similar speed gains, table 3 increases write power, gate driver breakdown susceptibility, and affects cells proximate to the driver to become 'over reset' resulting in 'stuck at faults'. An

Table 3. Case study for 512 × 512 array from [66] illustrating the resulting write latencies in figure 13.

Scheme	V_{write}	V_{dis}	W_{error}	P_{wr}	L_{wr}	E_{wr}
Single	2 V	1 V	Yes	—	—	—
Dual	1.6 V	0.8 V	No	1.74 mW	6.2 μs	10.7 nJ
Quad∗	1.4 V	0.7 V	No	1.46 mW	350 ns	511 pJ
Quad	1.2 V	0.6 V	No	890 μW	6.6 μs	5.9 nJ

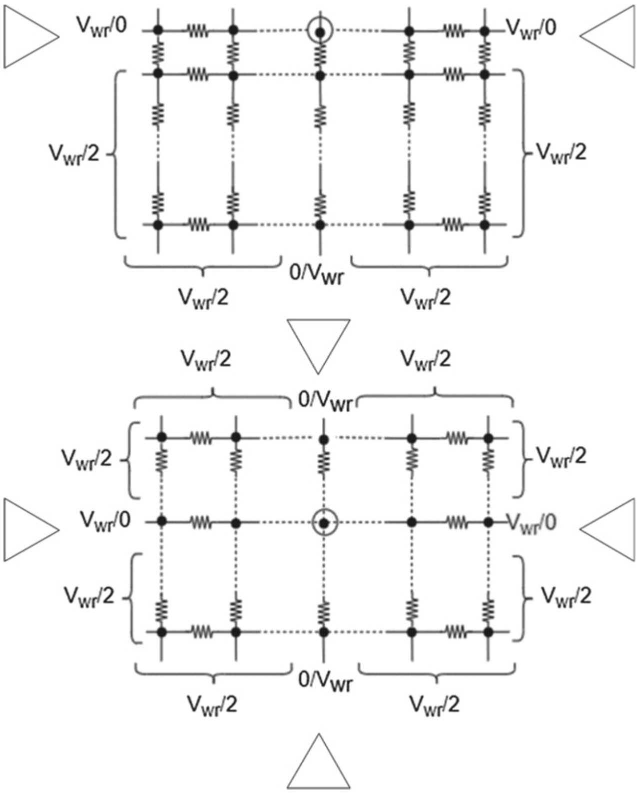


Figure 13. Dual (top) and quad drivers (bottom) scheme to reduced effective crossbar line resistance. © 2019 IEEE. Reprinted, with permission from author, from [66].

alternative angle to reduce wire resistance in [68] uses a double sided ground biasing scheme. The reduction of the longest IR drop path using this method means reduced latencies. In both methods additional drivers, and decoders are required. Also discussed in [68] is the data pattern effects on write latency 1–0 transitions versus 0–1 transitions.

4.1.2. Worst case latency

The switching time of an ReRAM crossbar depends on the array size, write current, wire metal resistance, and number of bits being written in parallel. ReRAM switching time is inversely exponentially related to its applied voltage, and the closer the selected cell is to the driver the shorter the switching time as it is getting the full write voltage. Further away due to interconnect IR drop and sneak currents, multiple ReRAM cells will see different voltage drop as illustrated in figure 12. A significant timing bottleneck to track is to ensure that the switching time of the furthest (worst case path) cell is less than the minimum reset/set latency [68]. The write latency of the furthest selected cell is proportional to $\tau \times \mathrm{e}^{KV_{\mathrm{d}}}$ where τ is switching time and K is a fitting constant [68].

4.2. Sneak path ‘crosstalk’ current mitigation

A major challenge for crossbar memory design is the interference from leaky currents in adjacent unselected cells which can cause write failures and misinterpretation in readouts. The sneak resistance can be modeled as a resistor in parallel to the desired cell resistance with the worst case scenario being when these unselected devices are in their lowest resistive states [69]. It is most commonly mitigated by using selection devices such as transistors to access the device in a 1T1R configuration or diodes in a 1D1R so limiting the current [70] or other novel means [58]. However, there is a penalty paid in the compromise between the selection device

Table 4. Ideal two-terminal selectors for emerging memories versus promising selector devices in development, information gathered from survey in [58].

Ideal 2 terminal selector	Insulator–metal transition (IMT) selector	OTS	CBRAM
Low off current	Higher than CBRAM and OTS due to structural defects in poly-crystalline thin film, along grain boundaries and at electrode—IMT film interface	Yes	Yes, if used well below compliance current. Uses the instability of CFs spontaneous rupture of the filament when bias is removed. At higher operating current this volatile behavior is lost and the device becomes more of a CBRAM memory type!
High selectivity	No due to the higher off currents	Yes, retains performance under stress conditions	Asymmetry in turn on and turn off as relies on self-rupturing to transition from on to off state
Infinite cycle endurance	Yes	No	No
Fast switching	Yes, and uniform switching in high-low, low-high resistance	Yes	No, slower than IMT and OTS—especially slow turnoff
Compatible operating conditions with intended memory device	Reasonable transition temperature for NbO_2 over VO_2	—	Need to choose operating current well below characteristic current for CBRAM volatile to nonvolatile transition while aligning with chosen synaptic device's operating currents
Compatibility with fabrication process (high thermal stability (need to be at $> 400°C$ to withstand BEOL process)	High thermal stability for NbO_2 based selector > 430 K	Poor thermal stability but shows potential in tellurium based OTS selectors by B and C doping	Thermal stability marginal due to metal ion diffusion at high temperature

conductance versus the synaptic resistance as this reduces the resistance window and hence number of multi-level synaptic states (especially when the selection device is too resistive). Conversely if not resistive enough, e.g. a leaky device, the selector cannot act as an effective current limiter. One selector-free architecture proposal is to raise driver voltages to overcome the sneak currents while another alternative is using fully-selected and half-selected cells, the latter's purpose to limit the amount of voltage drop in the non-active (half selected) cells [68, 71]. A more modular selector-free approach [72] to the problem of sneak-path is by reducing the crossbar into smaller modules and summing up currents from each of these modular crossbars prior to entering the activation unit burning more energy. A selector-free crossbar solution opens up higher density solutions for 3D growth representation of the resistive crossbar array. There are other means to achieve this as research is promising in the area of two-terminal selector devices [58]. The ideal selector requirements are listed in table 4 gathered from several works surveyed in [58]. Illustrated on the left of the table is the ideal two-terminal selector device and how two of the common types of these selectors match-up. Research is ongoing and there have been multiple means of mitigating these effects such as the high off current in the NbO_2 which make it unattractive for crossbar usage [73]. Similarly a means to improve thermal stability of ovonic threshold switching (OTS) devices is underway [74]. conductive bridge RAM (CBRAM)-type devices show great promise as a selection devices as long as the operating currents are well below its compliance current to allow it to remain in a volatile state *while* still aligning with the intended synaptic emulator device's operating currents.

This section looked at crossbar considerations for ML learning with analog neural networks. For overall system modeling, crossbar dependencies can be incorporated into crossbar modeling by extracting all the non-idealities in the crossbar and adding them into the aforementioned software ML frameworks (e.g. [39]) to create a fast crossbar model for ML evaluation [75]. The latter is a pseudo-emulation model with the conductance non-idealities pushed into the weight tables to model resistive crossbars. [67] proposes a flowmap for crossbar ReRAM based array configuration with input from driver finite resistance, application matrix technology node and ReRAM model to optimize the matrix mapping to the crossbar array.

5. Synaptic device candidates

Crossbar devices act as synaptic emulators for neuromorphic computing and allow for the co-location of computation and memory. These devices can be split into

- Volatile—most of these are charge-based (storing information in the presence or absence of charge), such as FLASH, SRAM, DRAM and
- Emerging non-volatile devices in these cases resistance based devices [1] which use a physical property that represents a conductance change (changes in device atomic arrangements or ferromagnetic layer orientation)—such as ReRAM, PCM, STT-RAM, FeFET.

Conventional NVM memory requires large ratios between the R_{HRS} and R_{LRS} states to allow reliable explicit readout of a binary value. With deep learning however, this window needs to have multi-state capabilities and the readouts to become more of an accumulation of multiple device effects for matrix vector multiplication. The primary focus of this paper will be on the emerging non-volatile resistive based devices over the charge based, conventional CMOS based memories (FLASH, DRAM, SRAM) which require a larger number of transistors thus are not area-efficient. In the case of FLASH also requires much higher operational voltages resulting in higher latencies and lower endurance due to gate oxide breakdown caused by larger electrical fields.

Synaptic weight updates can be positive or negative and with a physical device there are various means to realize this negative update. One means of doing this is to have two different conductance elements for each crosspoint so that the equivalent synapse is differential [1, 3]. This is especially important for unipolar devices where for example (PCM) set process is gradual while resets are abrupt so synaptic weight updates focuses only on the set process for positive and negative updates. The unipolar device is paired with another device with matched linearity and a reset strategy [3] is used to track saturation of one device over the other so restoring differential resistance and preventing network freeze-out. With bipolar switching devices that allow gradual change in conductivity for both sets and resets, linearity requirements can be more relaxed with preference to symmetry in set and reset [1, 52]. To create negative and positive updates architectures can use a local reference element the same for all rows and columns that sets the ‘zero’ threshold. Another method is to set the average value

$$\frac{R_{\mathrm{LRS}} + R_{\mathrm{HRS}}}{2} \tag{1}$$

setting as the zero weight setting [61].

5.1. Requirements of analog synaptic devices

The primary requirements for analog synaptic devices in the crossbar architecture are:

- High on/off ratio, the window between the applications usable device high resistance (R_{HRS}) and low resistance state (R_{LRS}). This defines whether the device can be used as a multi-level-cell (MLC) defining how many realizable conductance states.
- Weight update linearity and symmetry.
- Distinct R_{LRS} (on-state resistance, low resistance state) and R_{HRS} (off-state resistance, high resistance state).
- An accommodating average resistance, a smaller average resistance relative to crossbar interconnect means parasitic crossbar interconnect resistances dominate in IR drops.
- Reliable number of multi-bit states within the conductance window—R_{LRS} and R_{HRS} that will allow reliable gradual uniform conductance changes. With small windows a binary multibit solution, where multiple devices are placed in parallel is an alternative.
- Fast switching speed (low write latency).
- Fast access time (low read latency).
- Long retention time (non-destructive readout).
- High endurance for repeated programmability.
- CMOS process compatibility.
- Reduced cycle-to-cycle and die-to-die variation.

The requirements will vary based on primarily the deployment application (inference versus training), or edge versus cloud based applications. Further requirements breakdown into expected workload (image classification, NLP, object detection) which then affects the type of neural network architecture and depth. For example, a training solution will require multi-state devices, allowing linear gradual conductance updates, high endurance and faster programming speed than an inference solution because of the need to back-propagate involving several epochs of writes. A forward inference solution can have these at a lower priority favoring faster access time for reads, and higher retention devices where read-disturbance is limited. An inference solution would favor devices with one time programming/mapping to sustain non-disturb MVM reads for a prolonged period of time. The spider chart of figure 14 summarizes this and illustrates, for two popular NVM candidates [63], PCM, ReRAM, how well they meet these requirements.

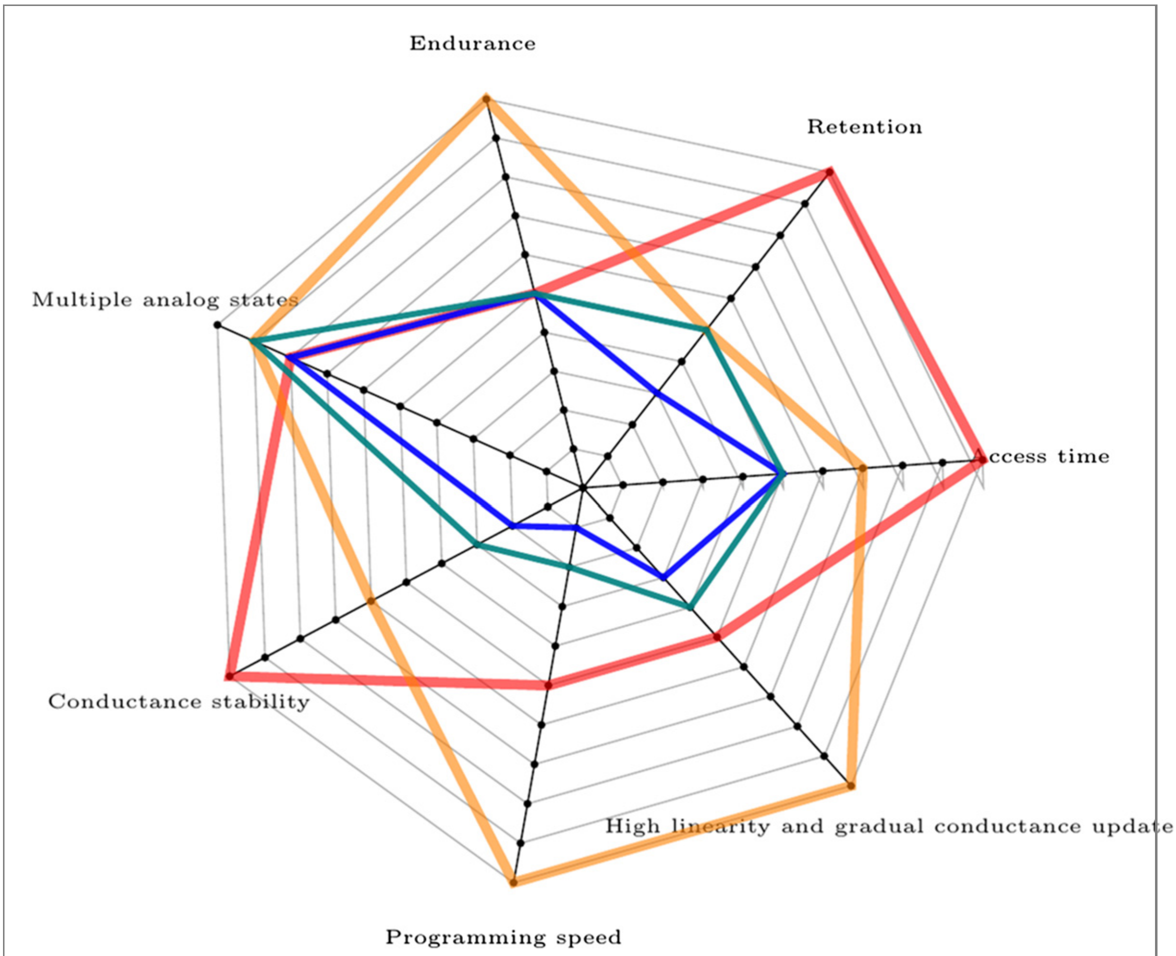


Figure 14. DNN analog device requirements for inference (pre-trained *ex situ*) (in red) and *in situ* training (orange), PCM (blue), ReRAM or CBRAM (teal).

An interesting part of the retention discussion is volatility and memory capacity effect where by memristors in [76] are not effectively trained due to the current limiter device, also the larger the parallel network the weaker the memory so there is a need to adjust the current supply limits to accommodate. These parallel devices (while allowing discrete multi-states and great for absorbing variation effects) means that the intrinsic conductance decay of the devices is more concentrated as there is now a competing natural decay thus penalizing retention even further. Volatility effects are studied as the RC decay time constant relative to the time needed for one epoch (forward propagation, reverse propagation and weight update) the higher this number is the greater the classification accuracies [6], this was studied on a 5000 examples of an MNIST data set using PCMO, ReRAM implementation. A low value means more volatile so need larger learning rates (retraining many more of the weights).

As discussed prior in section 3 conductance can be modulated based on the history of signals applied to the device and [77] looks into the variance of the synapse to the same pulse (width/amplitude) presented at different time stamps consecutively.

CMOS compatibility [52, 78] is important to reduce the number of fabrication steps and ensure memristor operational voltages are aligned to other circuit expectations. The prudency of ReRAM technology scaling results in higher programming voltages compromising other circuitry and the approach in [79] with a monolithic 3-D IC stack allows integration of two technology nodes at BEOL where CMOS peripherals are kept at more advanced nodes (16 nm). Each memristor-array 'tile' (40 nm) interfaces with the next through interior vias after processing through peripheral circuitry and logic (sense amplifiers activation pooling, buffering) in the 16 nm technology. The impact of the worse case latency scenario (single device activated in a column) through the interior via resistance is additionally investigated on the ADC sensing capability and illustrates minimal impact [79].

5.2. Device type and structure

Recent research has shown interest in ReRAM, PCM, STT-MRAM, FeRAM where multilevel programmability can be applied using electrical pulses. Also other devices like battery like, capacitor based, photonics are of interest to researchers. This section and paper will primarily focus on emerging NVM devices in 2D crosspoint arrays which have shown potential for neuromorphic matrix vector computations.

SRAM: in memory computing in CMOS technology using SRAM is possible but with limited density [28, 80], (with higher density NVM technologies are able to store multiple states in a $4F^2$ footprint). The SRAM cell is built from back to back FETs and two selectors (6T STRAM) with no dedicated storage element and charge needs to be constantly refreshed so needs to always be connected to a power supply [1].

DRAM: a capacitor acting as the storage node is placed in series with an FET and needs periodic refresh. The challenge for DRAM are the destructive reads and nondestructive attempts to overcome this cause degradation in density [81].

FLASH: in FLASH devices, the storage node is coupled to an FET gate and allows for longer term data retention but operating voltages are extremely high with large latencies. FLASH has lower endurance due to oxide breakdown from the large electric fields [82].

ReRAM: ReRAM devices are the most mature device candidates and are already being fabricated commercially [3, 83, 84]. They have strong compatibility with CMOS fabrication as they have BEOL compatible temperatures only needing one extra lithography step thus reducing costs. They also have long development history for learning applications [52, 85].

ReRAM can be split into filamentary and non-filamentary ReRAM devices [86]. Filamentary devices can be further sub-categorized into cation-based or anion-based, according to the means in which the conductive film is created [87]. In cation based devices (CBRAM) when a positive voltage is applied to the top electrode metal (usually Ag or Cu), metal ion oxidation occurs where the anions are attracted to and collected onto the opposite relatively inert electrode. The buildup of these anions with continued applied voltage will eventually form a conductive path between the electrodes. With anion based devices (HfOx, TaOx, TiOx) however, the conductive filament (CF) is gradually formed through the metal oxide electrolyte insulator from the migration of oxygen vacancies through the electrolyte shown in figure 15. In the case of non-filamentary, the electrodes metal atoms form the conductive connection through oxygen vacancies [86]. Filamentary ReRAM (CBRAM)—exhibits low programming energy, fast switching, and high endurance but high resistance window (100×) and intrinsic variability [3]. This is compared to non-filamentary RAMs smaller resistance window of up to 50×.

Electrical pulses induce the set processes, associated with CF formation, and reset process, associated with the dissolution of the CF. If both processes are in the same voltage polarity then it is a unipolar process and if at different voltage polarities then it is bipolar [88]. To control the multilevel states gradual dielectric breakdown is achieved by controlling the number of CFs/controlling the amount of oxidation [52]. Bipolar filamentary RAM sets are usually abrupt versus gradual resets thus calling for a 2-ReRAM synapse differential readout approach like the PrCaMnO devices in [6]. Another option for a device that does not show gradual conductance change is as in [89] where multiple 1 bit/binary BNOx memristors are integrated in parallel to create a compound synapse thus representing a multi-bit solution. With all devices inclusive of those that do show uniform gradual conductance change, 'single shot programming' is not possible to precisely set the conductance level [70, 90] but a series of pulses. [52] has shown promising gradual bidirectional programming abilities that allow for incremental resistance changes with voltage pulsing.

The number and size of CFs can vary creating variations from device to device and cycle to cycle and [88] mitigates this by the use of buffer layers to confine CF paths. Changing the compliance current can also be used to alter the diameter of the CF. To ensure CF formation, electroforming or 'priming the oxide' for OxRAM [52, 70] is used where a large electric field ($>$10 mV cm^{-1}) is applied and causes soft dielectric breakdown creating defects in the oxide allowing CFs to form during sets. Reliance on electroforming to form the conductive paths allows for lower driver voltages during set operations which also avoids gate oxide breakdown of driver gates, (the larger deep gate oxide gate alternatives being slower). While forming enables the device to be controlled by smaller driver voltages to achieve the same resistance, it compromises the memory window ($R_{\mathrm{HRS}}/R_{\mathrm{LRS}}$) as while R_{LRS} are reduced, the R_{HRS} are also reduced as compared to the relative resistances of the initial fresh samples. A forming algorithm solution is presented in [91] which allows certain devices already preformed by the anneal process to be skipped thus avoiding further device-to-device memory window variation.

The exponential dependence of current on applied voltage can be expressed [92] as $I(d,V) = I_0 \exp(\frac{d}{d_0}) \sinh(\frac{V}{V_0})$, where d is the gap size between the CF filament tip and the electrode, I_0, d_0 and V_0 are fitting parameters. The linear range of the IV characteristic curve for the responsive devices of a 128 × 64 ReRAM array down to a precision of 6 bit (64 levels) is illustrated in [93] an additional data point to explore would be the temperature dependency of this curve [94]. A similar plot of the effect of linear range using the differential conductance provided in [90] that uses the 2-ReRAM synapse approach for reducing cycle-to-cycle and device-to-device variations.

The transition from short term memory (STM) to long term memory (LTM) is discussed in [95]. With repeated stimulation, the CFs become stronger as there is a higher concentration of oxygen vacancies in the

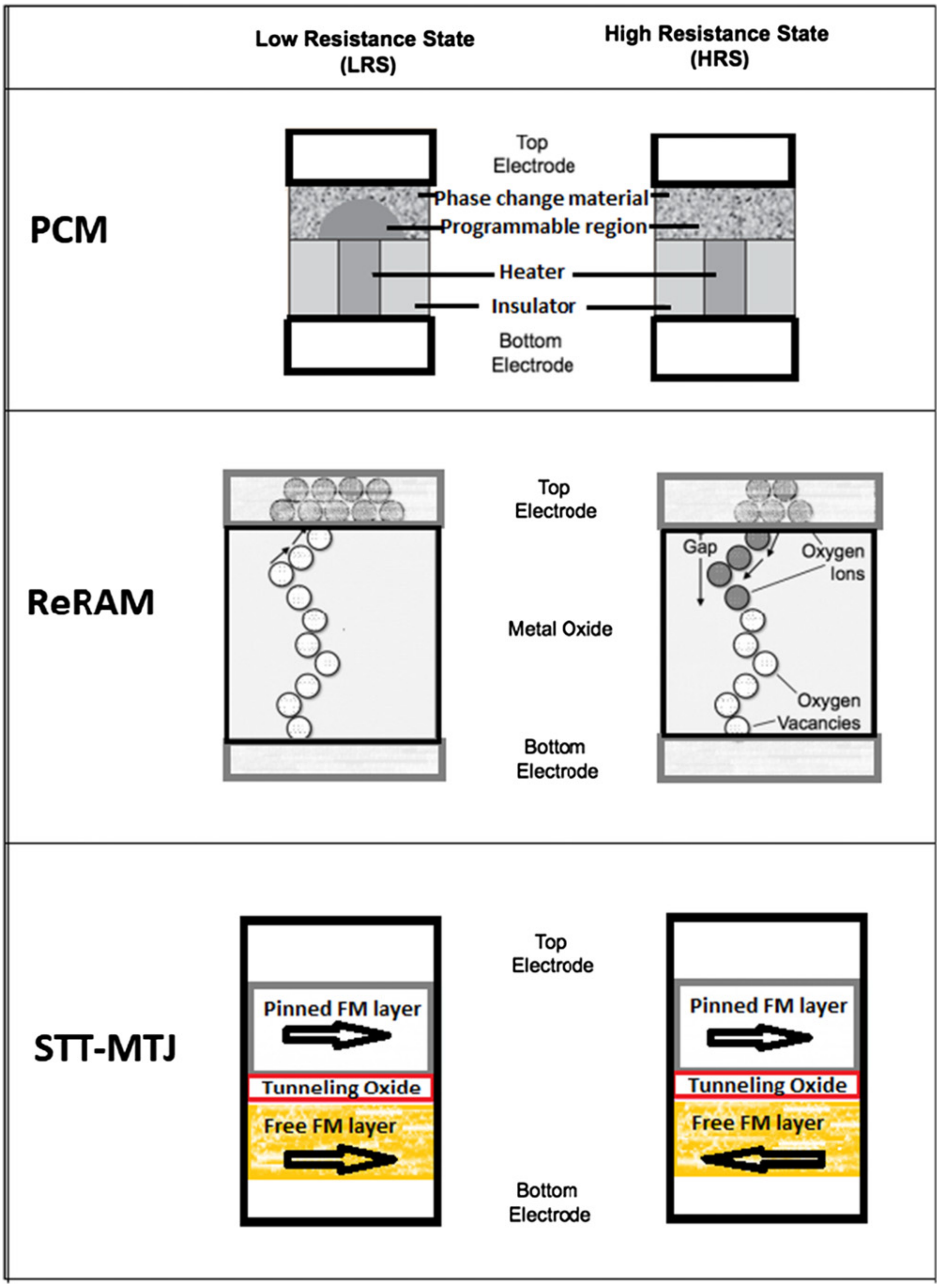


Figure 15. Two-terminal NVM technology devices: PCM, ReRAM, and STT-RAM.

switching layer and more resistant to lateral diffusion to break conductive path and thus resulting in higher retention. Diffusion of ions in conducting channels causes decay in retention LTM versus STM. Also to consider is stuck-ats/device unresponsiveness [88, 96] where stuck at R_{LRS} and unable to reset to R_{HRS} occur due to too many defects in the switching layer. Several architectural means of avoiding and building redundancy into the network can help and are discussed in section 7.

PCM: the second of the leading choices for analog based accelerators is the PCM. This two terminal chalcogenide, out of the listed NVM storage class emerging memory candidates, has the highest on/off ratio second only to 3D NAND FLASH [98]. Its amorphous phase exhibits high electrical resistivity while its crystalline phase shows low resistivity about several tens of orders of magnitude lower [55]. This opens up the space for multi-level cell operations. The amorphous phase is an abrupt melt-quench process that is initiated by a large amplitude short voltage pulse while the crystalline phase is when material is heated using lower amplitude longer pulses. Due to this, to realize the different multi-level states (in both directions) gradually changing amorphous thickness with progressive crystallization [63] through controlled heating (electrical pulses) of the chalcogenide material is required. The opposite direction, incremental reset of PCM is not possible because of the abrupt nature of amorphization, ‘reset process’, so similar to the filament based ReRAM (with its abrupt

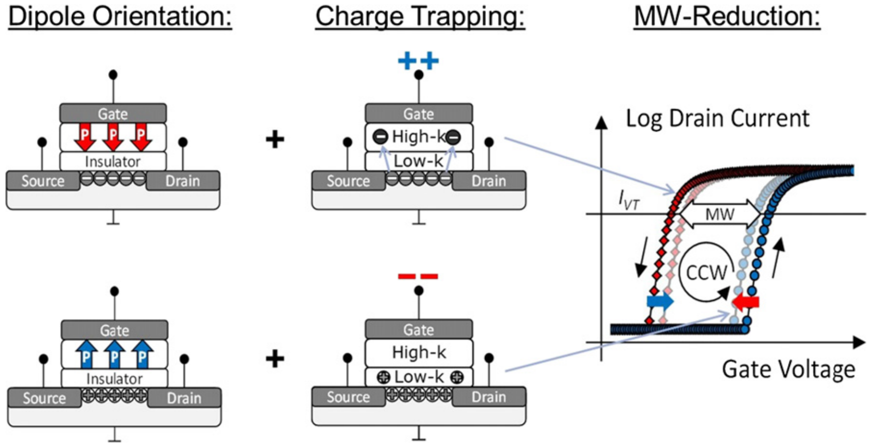


Figure 16. FeFET ferroelectric polarization and eventual charge trapping. With increased trap density over the operating life the memory window (MW decreases). © 2016 IEEE. Reprinted, with permission from author, from [97].

set [6]) using a pair of devices as an equivalent weight to represent positive and negative weights and mitigating asymmetry in set/reset [3] is required. In the crystalline state, the PCMs show ohmic dependence at lower voltages and non-ohmic when voltages are higher. The large currents needed to write a PCM cell [68] limits the number of parallel writes [89] as the crossbar needs to stay within electromigration limits.

Another challenge of PCM devices is resistance drift that is caused by spontaneous structural relaxation after the melt-quench process, conductances initially decrease rapidly then more slowly [6]. This is studied in [99] where the change in relaxation is investigated over time and temperature (considering also array level impacts where arrayed devices exhibit different drift components). $G \propto t^{-v}$ where t is time, G is conductance and v is the drift coefficient [6]. With strong resets where cells are fully-amorphous, drift components are larger thus affecting data retention and hence network classification accuracies [63].

STT-RAM: an NVM two terminal device based on magnetic materials that has been widely studied for neuromorphic applications, due to its promise of high density and low leakage, is the STT-RAM [43, 63] that uses electron spin to store resistive state. An metal tunnel junction (MTJ) is created by a spacer between two ferromagnetic (FM layers); one layer called the free layer and the other a reference layer. The relative orientation of these layers is controlled by passing a current to each FM layer to either have a parallel or anti-parallel direction to create the resistive states. For the R_{LRS} a current is applied from the reference to the free layer so that the magnetic orientation of the free and reference layer are the same, this is referred to as ‘parallel’. The opposite would be used to realize a logic 1, or R_{HRS}. This is how a single bit cell or single level cell works. To extend to a multilevel cell this would require stacking of differently sized MTJs [100], one challenge is the low tunneling magnetoresistance [63], as well as reliability problems with process and thermal fluctuations in the MTJ. Write reliabilities can be improved by using higher currents and results in faster switching times but could adversely affect reliabilities for MLCs as several MTJs are in consideration. Researchers have looked at techniques such as early write termination, hybrid SRAM/STT-RAM architecture and read-preemptive write-buffer designs [68] to mitigate the long write latency of STT-RAM. Reading has its challenges as with the smaller $R_{\mathrm{HRS}}/R_{\mathrm{LRS}}$ ratios the distinction between the states becomes challenging and coupled with thermal fluctuations, worsens read disturbance effects.

FeFET: while the aforementioned NVM cells have two terminals, the FeFET is a three-terminal transistor device acting as its own selector thus allowing for more compact memory arrays [101]. It is an MOSFET with a ferroelectric gate dielectric (commonly HFO_2 based). The cell has two distinct stable polarization states and can be switched using an external electric field (‘coercive field’) the strength of which determine the extent of polarization as each crystal domain within the structure is polarized. The remnant polarization after the electric field is removed allows data storage through these two polarization states. The two states are referred to as a low threshold (low V_{t}) and high threshold (high V_{t}) states and the memory window is defined as the difference between these two stable states. With the aging device the memory window closes due to charge injection from the substrate due to wearing of the thin film interfacial layer between the ferroelectric dielectric and the silicon substrate shown in figure 16 [97]. Various means to reduce this are discussed including changes in process flows [102], use of a series resistor 1FeFET1R (1F1R) to reduce V_{t} variation [103]. Research is on going in improving device to device variation, endurance and increasing the memory window for multi-level performance [104].

5.3. Device reliability

In the section 4 the effects of crossbar non-idealities were discussed, from wire interconnect, source and sink resistances that create linear and non-linear idealities when it comes to read accesses and thus output current inaccuracies. Discussed in this section are the contributions from device behavior which for example in the case of PCM whose IV characteristics at lower voltages displays ohmic behavior to exponential-behavior at higher voltages. In contrast, ReRAM current is exponentially dependent on voltage. In addition, access devices also play a role in this as they are in series further contributing to inaccurate read process—if not efficient current delimiters thus inaccuracies in read output current and/or affecting available resistance window for MLC use. Careful consideration is thus required for voltage ranges for read and write pulses. Pairing devices to overcoming asymmetries in set/resets was discussed [63] and can be extended for PCM and ReRAM. In an offline trained NN, device non-idealities can be overcome as conductance values can be programmed reliably by doing a write–verify–write offering opportunities to correct [105]. In the case of online trained (*in situ* training) however, it is paramount that the conductance updates are symmetric and linear.

Larger crossbars mean higher impact of interconnect resistance overcoming the presented effective device resistance thus impacting accuracies, but limiting the crossbar size to a small size means fewer errors but more power as more crossbars are needed to represent each layer. Lowering the average device resistance $(R_{\mathrm{LRS}} + R_{\mathrm{HRS}})/2$ has the similar effect to increasing crossbar size as this means greater impact of interconnect parasitic so higher average values are preferred to reduce parasitic impacts. A small $R_{\mathrm{HRS}}/R_{\mathrm{LRS}}$ ratio also means few bits per device and lower area efficiency.

Reliability effects on ReRAM technology are studied in [70] where the CF growth of PMC devices show high tolerance to ionizing radiation exposure. CF rupture shows less tolerance (and thus R_{HRS}) is slightly higher than non-exposed devices. This means exposed parts have a higher $R_{\mathrm{HRS}}/R_{\mathrm{LRS}}$ ratio. Exposure however, has little impact on retention for both resistance states up to a total ionization dose of 2.6 Mrad opening up usage in more environments.

Time dependent variation [3] is more pronounced at R_{HRS} states so during backpropagation, accuracy can be affected. Higher endurance is needed for small conductance changes instead of large changes in digital memory applications. With the multiple conductance update steps in backpropagation (and asymmetry between increasing and decreasing conductance) meeting convergence becomes challenging. Several tools [75, 106, 107] are used to model NVM based networks and evaluate system performance, they provide some direction on circuit area, leakage power information, latency and energy consumption.

Random initialization to break symmetry [90] aids in convergence and is easily provided due to intrinsic device-to-device variation. Another suggestion is to additionally assign the memristors somewhere in the middle of the conductance range [108] and within the useful section of the squashing function. Further, [109] discusses how to locate and initialize memristor synapses as this initial value is argued to be affect memristor variation. Several models are proposed to describe the memristor behavior and initial state, a mapping simulator software to map DNN to resistive crossbar to aide in the analyses [110]. A detailed look at weight initialization and distribution method [3] from centroid initialization, to random, density based and linear is discussed in [111] and can provide some application for NVM-based weight initialization and weight quantization binning [112]. Similar applications are used in mapping from offline-trained weights onto the ReRAM crossbar [3, 113].

Several means of extending the life of the device and improving retention and endurance are discussed in various research such as ‘periodic carry’ [63] where a set of parallel devices represent a single synapse, each having a different weight toward the total synaptic conductance. So when LSB saturates to its max or min, the next least LSB is updated to account for the information from the previous LSB, (while the original LSB is then ‘reset’ away from its saturated value). This technique avoids ‘overuse’ of a single device thus extending usage. A take on the ‘periodic carry’ concept is also presented in [114] where a different device is introduced for the lower significant device pairs. The only demands for this device being high linearity for conductance updates, and high endurance thus protecting the larger more significant NVM-based device weights from ‘overuse’ degradation and relaxing their linearity requirements. A similar option to prevent overuse and hence extend device lifetime and average variability in devices is to put multiple conductance of equal weight contribution and update is done by programming one at a time to reach the conductance step needed. An arbitrator timer will make sure that they all get a similar number of requests to avoid saturation of one device or have endurance failure of one NVM. So, each device is only programmed once per several updates. The latter can also be extended to form a single equivalent synapse conductance made-up of several NVMs in parallel to distribute variation effects [54]. A hybrid structure of different combination of memory devices to extend the conductance range and improve linearity of weight updates could also be considered, for example suggested in [63] is a PCM for MSB, and transistors to cap for LSB thus relieving training on PCM due to its high write latencies and resistance drifting. After training the final conductance value can be scaled and stored on the

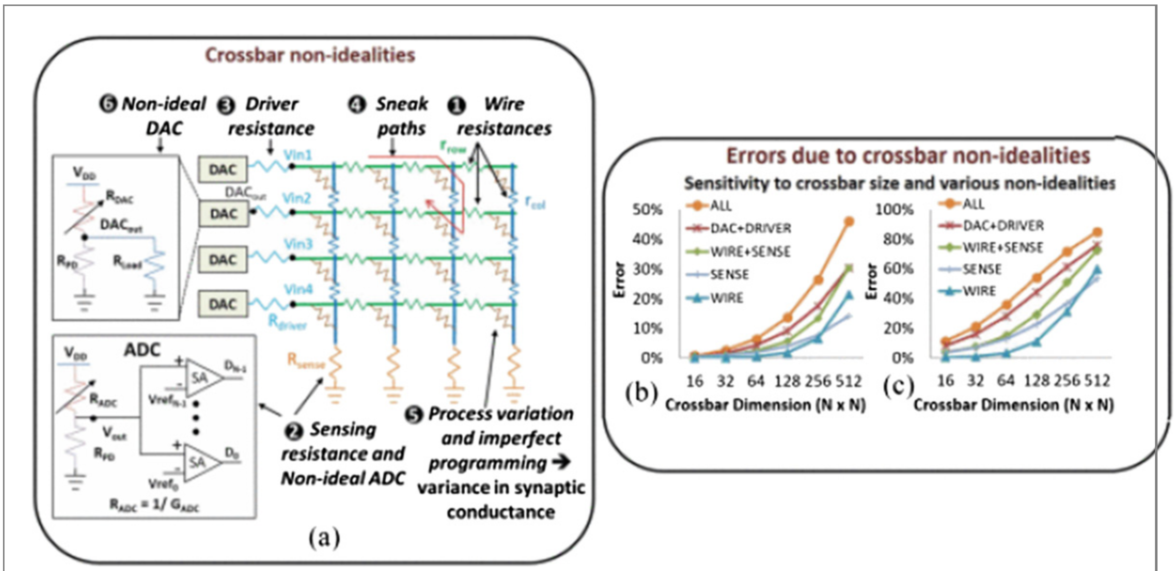


Figure 17. (a) Equivalent circuit of crossbar with data converters and contributors to errors. (b) Synapses programmed to G_{MIN} sensitivity to crossbar dimensions. (c) Synapses programmed to G_{MAX} sensitivity to crossbar dimensions. © 2020 IEEE. Reprinted, with permission from author, from [75].

PCM device. This hybrid nature can be used on other NVMs to take advantage of positive contributions a device has for DNN processing (see spider plot in figure 14).

6. Peripheral circuitry support

This section will discuss supporting circuitry and design considerations for the cross-bar array based neural network implementation. Support circuitry discussed in this section include DACs, ADCs, drivers and sensors. Additional structures such as multiplexers, switches, various circuit sigmoid implementations, sign control and weight update circuits are featured in [72] with accommodation for binary neural networks (BNN) which avoids the prolific use of power hungry DACs and ADCs. Major considerations for robust design center around area efficiency, low power, and precision/resolution.

6.1. Data converter circuitry

DACs: DACs are needed to drive analog voltages to rows or columns to allow forward and backpropagation in the crossbar based neural network. In both cases the DAC driven drivers should have the ability and range to drive read and write-update voltages to the synaptic memory cells (figure 17). Encoding architectures using time encoding tend to be low-speed as *several cycles* are required to generate the various pulses to effect reads or write updates in the synapse.

Investigated in [75] is the error from DAC non-idealities and how a DAC output voltage can change with average equivalent synaptic load resistance (R_{load} should also include effect of wire resistances) and is also a function of the applied input. Also illustrated in [75] and in figure 18 is that the sensitivities to crossbar dimensions is greater when including DAC and driver non-idealities thus resulting in the largest contribution (especially for crossbars less an 512 × 512) toward classification errors than sensor or wire non-idealities. This is due to the nonlinearities of the coupled with lower effective drive voltages in larger crossbars.

ADCs: ADCs tend to be area and energy intensive consuming up to 80% of total crossbar energy and about 60% of total cross bar area, the former increasing with the amount of precision required [63, 115]. One means to increase power and area efficiencies is the use time multiplexing to share the ADC across multiple columns [46] but this results in reduced throughput due to reduction in parallelism. Another is reduced precision, some studies have shown higher tolerances to accuracy degradation when ADC precision is reduced [116]. With *in situ* training however, this may not be a reliable knob as these solutions must prioritize the precision (and range) of neuron computations and subsequent activation [5] from each neuron to be supported by the hardware while offering a fast ADC response. Turning precision into a hyper-parameter knob for each neural network layer may regain some energy savings while preserving classification accuracies. BNN which allow for faster inference times (where MACs become bit-wise operations) and faster updates are only two levels [R_{LRS} (on) and R_{HRS} (off)] circumvent the need for ADCs but lead to accuracy degradation over time [72].

It is clear that fast ADC responses are needed to propagate through the various layers and time multiplexing of ADC architectures across various rows and columns compound this need. A simple high speed option is

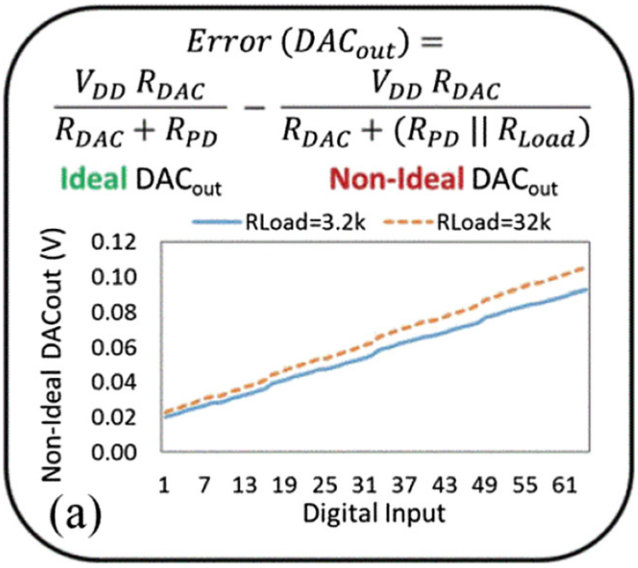


Figure 18. Non-ideal DAC output due to R_{load} (effective load resistance of the crossbar array). © 2020 IEEE. Reprinted, with permission from author, from [75].

offered by FLASH ADCs but with the large loading to the input voltage and the amount of area and energy required for all the comparators (an 'n' bit ADC requires $2^n - 1$ comparators) this means limited resolution trade-off with area. Most ML solutions suggest up to 8 bits resolution, (or 255 comparators!), which is not an area or power efficient solution. Also, with the increasing resolution of these architectures the difference between adjacent voltages become smaller than the individual inherent comparator offsets. Pipelining ADCs provide a solution but are limited to the number of states to reduced buildup of error from mismatch of the internal DAC stages and residual amplifiers. A common compromise approach in analog DNN studies is the use of SAR ADCs [44, 46], which presents lower capacitance loading to its input stage, higher resolution and lower power, but at the cost of lower conversion speeds (limited by the internal comparator and DAC speed divided by the required bit resolution). A new scheme providing superior power delay product than SAR and FLASH is investigated in [117] using an analog shift-add ADC scheme to do the weighted sum for up to six bit precision and comparable area to the SAR ADC.

In addition to conversion speeds, designs need to factor ADC settling times and time to latch outputs for further post processing (digitally handled RELUs, activation storage and pooling). One means of circumventing this is to use interleaved ADCs where multiple ADCs are interleaved in parallel with clock staggering and then outputs time re-aligned. This however means more loading to the input signal, additional clocking circuitry, and additional care to the non-ideal interleaving effects such as clock distortions, phase errors, mismatched ADC core offsets and gain errors—each of which will have their own correction techniques incurring further area and energy.

Research in [46] discusses the read ADC pipeline and reduction of overhead by sharing ADCs in an IMA (*in situ* multiply accumulate) cell that multiple crossbars share and creating a 1.28 GSps ADC unit to sample the 128 bit line current from its 128 × 128 crossbar. Also proposed in [46] is a method of copying common multiplication algorithms by splitting e.g. 1 bit computation into 16 cycles in order to keep high precision but limit DAC and ADC size to n bits ($n < 16$) by instead of having a voltage level being represented by the 16 bit value instead uses a stream of levels. These are then accumulated in an output register after the ADC. This means 16 cycles are required to complete the 16 bit input. Reduction in cycles can be cut in half by splitting the computation into different crossbars: one crossbar for 8 bit MSB and one for 8 bit LSB.

Conversely, an ADC free scheme of sensing a PCM cell resistance with up to 8 bits precision by dynamically changing the reference levels to achieve reduced access latencies to 5 μs is proposed in [118].

6.2. Drivers

DAC circuitry are usually modified for multiple functions as seen later in the architecture section 7. Prior to the driver and DAC, the type of encoding required for the particular architecture has to be decided depending on the device characteristics and intended operating range on the device IV curve. Whether the stimuli will be amplitude or pulse width modulated and at what amplitude levels and pulse duration, this also includes considerations for load versus driver resistance characteristics discussed earlier in section 4 and power constraints. Several categories of driver circuits can be considered, voltage mode versus current mode type drivers—these

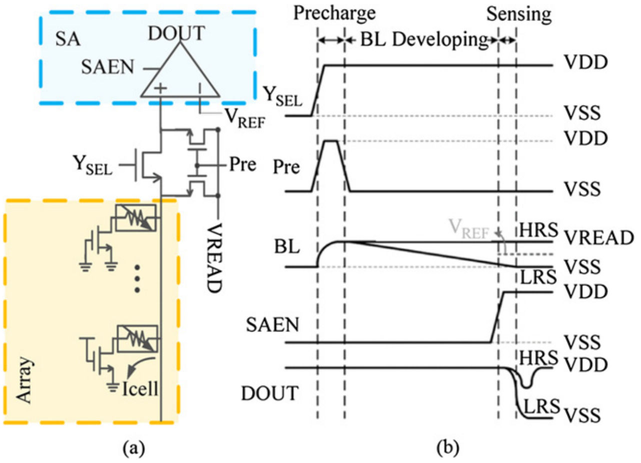


Figure 19. (a) Read path and (b) conceptual rendering of waveforms produced by nominal voltage-mode sensing scheme. © 2015 IEEE. Reprinted, with permission from author, from [124].

can borrow ideas [119–122] that allow for amplitude swing adjustment and slew rate control to efficiently pulse NVM devices (the latter feature an added plus to control crosstalk). An impedance control within the driver can also allow for dynamic crossbar dimension configuration [65, 119, 120]. Methods of reducing effective cross bar wire resistance to the driver were discussed in section 4 by increasing driver voltages or use of multiple driver configurations [66, 68] to reduce latencies.

6.3. Sensing circuits

Accumulated currents at the end of each crossbar column (or row in the case of backpropagation) will need to be sensed prior to digital conversion and subsequent storage or further processing (e.g. digital activation or pooling). The resulting accumulated currents are converted to analog voltages (voltage sensing) or currents sensed (current sensing) using various means [122–125]. In the former case the drop in bit line voltage is sensed versus a reference voltage after a pre-charge and development phase. In the latter, the cell current is compared versus a reference current generated by reference/dummy array (sometimes with dynamic clamping of the bitline (BL) for a faster precharge phase) as in the figures 19 and 20. A simple circuit diagram of the voltage and current mode comparators are shown in figure 21. The choice of sensing mode is dependent on the size of the array—specifically the amount of loading on the BL as more cells per BL (and higher R_{LRS}) means a longer access time as shown the figure 22. With the large BL loading (larger R_{LRS}) and long BL lengths it is best to choose current sensors for faster accesses. Challenges to current mode circuitry are variations in reference current which can cause read failures when overlaps with read currents occur. Fluctuation in the BL clamp voltages means fluctuation in the voltage drops across the memristors. The voltage mode also has its challenges: as memristor variations result in a wide range of BL voltages so there is a need to select the reference to accommodate and/or schemes to track accordingly. Lower supply schemes have their effects in both cases, with current sensing the headroom of clamping device is compromised and in voltage mode sense circuits the lower voltage drives cause longer access times. Data pattern also affects access times as the different RC delays are presented to discharge change depending on how many memristor cells are selected—in [124] a simplified RC model of the bit line illustrates how the bit line discharge time increases with the percentage of R_{LRS} and discusses various techniques to mitigate these effects.

Speed is key in sensing circuits [126, 127] as this allows for propagation to the next stage and/or muxing to share sensor circuitry with other crossbar columns. Proposed in [123] is a proposed low latency current sensing technique and the effects of crosstalk and supply noise.

In a TIA circuit where current is integrated on to a capacitor and then sensed, integration time of the capacitor is based on acceptable noise tolerance of the integrator circuity and on/off ratio of the synaptic emulator [2, 128]. Noise tolerance can be increased by increasing this integration time albeit adding to more latency

$$C_{\mathrm{int}} = 2N\frac{V_{\mathrm{in}}}{V_{\mathrm{out}}}\left(\frac{\beta-1}{\beta+1}\right)\frac{t_{\mathrm{int}}}{R_{\mathrm{dev}}}.$$

Where R_{device} = average device resistance, N = the number of contributing devices, β = the ratio of $R_{\mathrm{LRS}}/R_{\mathrm{HRS}}$, V_{out} = voltage at output of the op-amp. With decreasing integration time more throughput (operations/second) can be achieved.

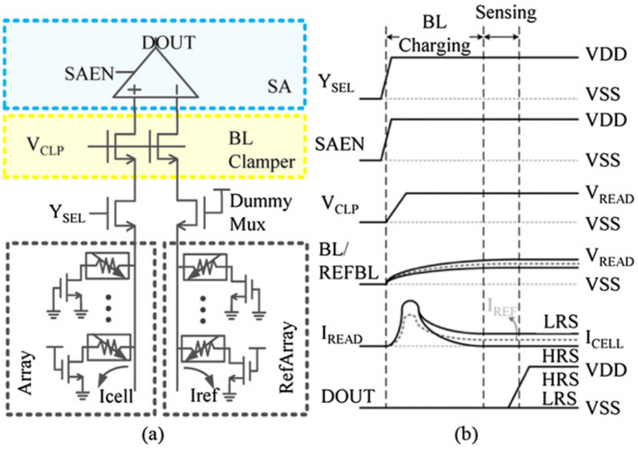


Figure 20. (a) Read path and (b) conceptual rendering of waveforms produced by nominal current-mode sensing scheme. © 2015 IEEE. Reprinted, with permission from author, from [124].

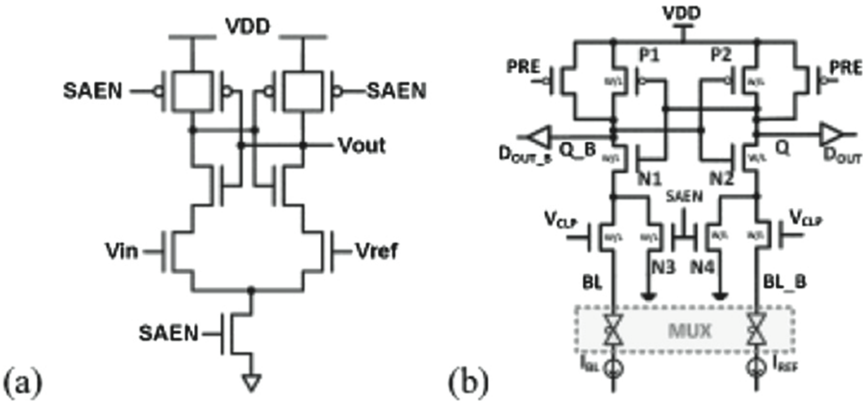


Figure 21. Example of (a) simple voltage-mode comparator (b) simple current-mode comparator. © 2015 IEEE. Reprinted, with permission from author, from [125].

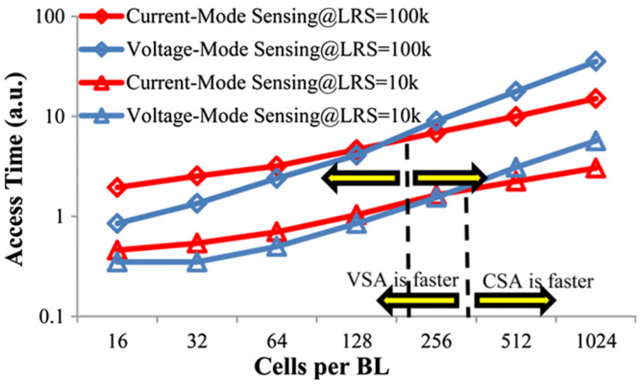


Figure 22. Access time versus BL length for common voltage-mode sense amplifiers and current-mode sense amplifiers. © 2020 IEEE. Reprinted, with permission from author, from [124].

6.4. Activation function and derivative circuitry

Circuit approximations of neural functionalities to drive reduction in area and complexity are studied in [5, 6, 51]. Discussed in the implementation is the replacement of the sigmoid activation unit implementation such as a tanh/ReLU since they require high precision A-to-D and D-to-A circuitry with approximation circuitry as a PWL (comparator and ramp voltage). Similarly for backpropagation of correction errors the MAC sum will need to be scaled by the derivative of the activation unit. The derivative of the PWL, which is a step function is used by specifying two distinct states of the step function. Illustrated in [6, 51] shows that

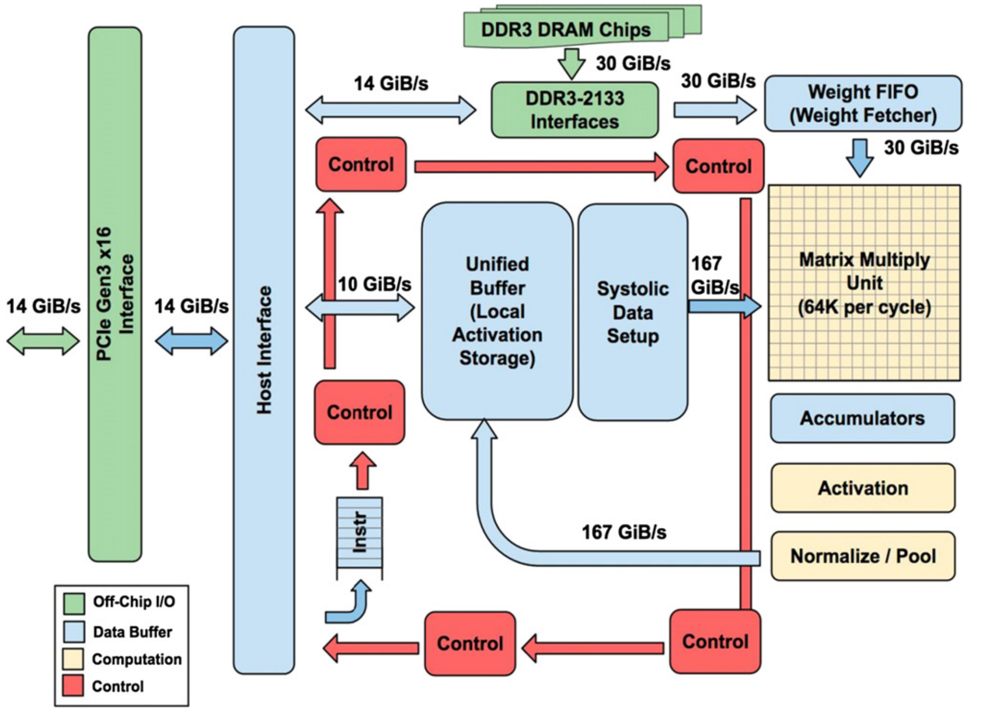


Figure 23. Tensorflow processing unit architecture. Reproduced with permission from [30].

on a 60k MNIST data set the training test accuracies of tanh() and PWL activation functions are comparable and can be further improved when changing the derivative of the low value of the approximation function to a non-zero as discussed in section 3.

6.5. Other support circuitry

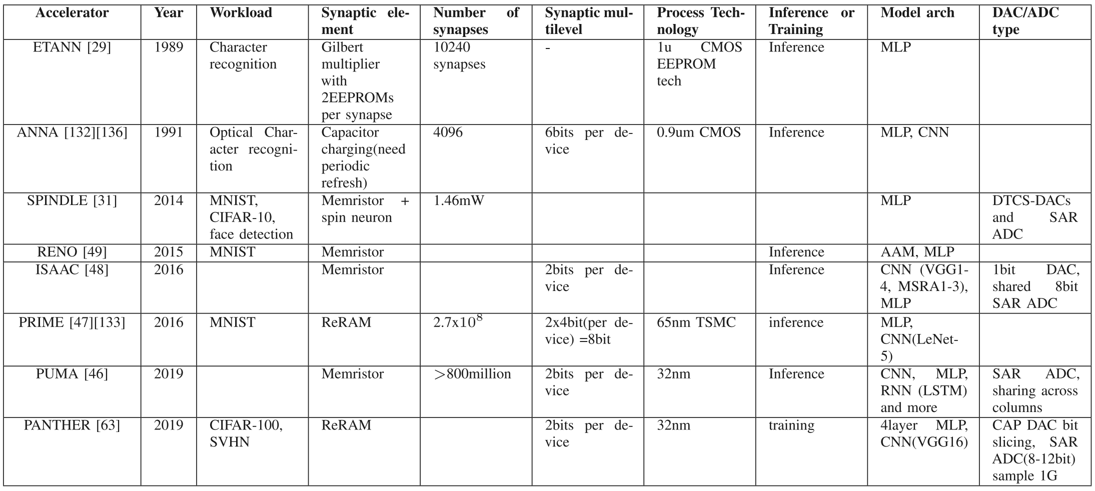
It was mentioned earlier that write voltages for emerging NVMs (PCM and RRAM) are much higher than the logic supply voltages (and with advanced nodes are more significant challenge [79]) so there is a need to make provisions for high voltage supplies, level shift circuitry and charge pumps [125].

7. Architecture and system considerations

A high-level view of architecture processing unit from Google's TPU ASIC [30, 33, 36] is shown in figure 23. The custom ASIC fetches weights from nearby DRAM and inputs through the high bandwidth memory (HBM) interface through the matrix units (MXU) for multiplication and subsequent accumulation before activation, normalization, and pooling before being written back into the HBM for use in the next layer. There can be multiple instances of MXU in each core and multiple cores within the ASIC. Other current commercial accelerators follow a similar architecture [42]. Analog based accelerators will have to have different architectural approaches due to the NVM crossbar based neural networks and so must provision for:

- Avoiding weight saturation [3, 5, 61, 129],
- Enhancing weight endurance and retention [56]
- Synapse weight inline calibration [47]
- Dual polarity based weight programming [5, 130, 131]
- High precision techniques (especially for training applications [61]) while keeping ADC overhead at a minimum [29, 46, 121, 122]
- Pipelining to minimize hazard conditions and reduce buffering from one layer to the next
- Synapse suppression to allow for structured sparsity
- A solution to not only avoid but accommodate 'stuck at' faults [132]
- Network flexibility to dynamically reconfigure network shape and size [5]
- Network pruning for sparser representation of the crossbar (even for FC layers) [4, 113, 133].

The aforementioned requirements have to be all coordinated by a robust instruction set architecture.



Accelerator	Year	Workload	Synaptic element	Number of synapses	Synaptic multilevel	Process Technology	Inference or Training	Model arch	DAC/ADC type
ETANN [29]	1989	Character recognition	Gilbert multiplier with 2EEPROMs per synapse	10240 synapses	-	1u CMOS EEPROM tech	Inference	MLP	
ANNA [132][136]	1991	Optical Character recognition	Capacitor charging(need periodic refresh)	4096	6bits per device	0.9um CMOS	Inference	MLP, CNN	
SPINDLE [31]	2014	MNIST, CIFAR-10, face detection	Memristor + spin neuron	1.46mW				MLP	DTCS-DACs and SAR ADC
RENO [49]	2015	MNIST	Memristor				Inference	AAM, MLP	
ISAAC [48]	2016		Memristor		2bits per device		Inference	CNN (VGG1-4, MSRA1-3), MLP	1bit DAC, shared 8bit SAR ADC
PRIME [47][133]	2016	MNIST	ReRAM	2.7x10^8	2x4bit(per device) =8bit	65nm TSMC	inference	MLP, CNN(LeNet-5)	
PUMA [46]	2019		Memristor	>800million	2bits per device	32nm	Inference	CNN, MLP, RNN (LSTM) and more	SAR ADC, sharing across columns
PANTHER [63]	2019	CIFAR-100, SVHN	ReRAM		2bits per device	32nm	training	4layer MLP, CNN(VGG16)	CAP DAC bit slicing, SAR ADC(8-12bit) sample 1G

Figure 24. Analog based concept accelerators.

A chronology of analog based accelerators is discussed in figure 24 and their features are described in the following text. Some are architectural concepts [44, 61, 135] based on a particular technology node while others are full [27, 28] or partial implementations in silicon. One of the earliest takes of an analog based accelerator the ETANN [27] uses EEPROM driven floating gate devices to store and adjust its 10 240 synaptic weights, and Gilbert multipliers are used as multipliers and routed, after current summation, to a sigmoidal function emulator. This is a static architecture dependent on calculation of weight changes and voltages externally to be applied to modify the weights.

ANNA [130, 134] (0.9 μm CMOS process) uses capacitive charge refreshed by external RAM to store synaptic weights for optical character recognition application. Additionally, as in ANNA, many of the architectures that follow [29, 131], the data converters provide combined functions—the DAC serving also as a multiplier to multiply a charge driven weight bias with the digital inputs. The large voltage range on the capacitance is to minimize errors due to charge leakage while a refresh circuitry is provided to compensate for this leakage. A means of handling the positive and negative weight contributions is provided within each multiplier cell. The SAR ADC not only combines a current comparator to compare the sampled signed summed current to a reference but also provides a squashing function characteristic to form the neuron body circuit. The overall ANNA [130, 134] architecture provides several orders of magnitude speed advantage over conventional hardware in use at the time.

Fast forwarding over 13 years later with growing interest in in-memory computing to overcome the von Neumann bottle-neck TrueNorth [28] (arguably analog based) was built on a 28 nm process and provides 256 million synapses and 1 million neurons with a neurosynaptic SNN core network. It provides more flexibility and scalability than its predecessors due to its tiled crossbars and provides time multiplexing between its core, a feature that continues with subsequent accelerators [29, 44–47, 61]. It is an inference only application where weights are trained offline and transferred onto an SRAM array for forward propagation.

Near computing approaches in a 28 nm concept model DaDianNao [135] are discussed using synaptic weights from adjacent EDRAM banks to the computational units (each tile containing 4 EDRAM banks), but, in this architecture strategy, *neuron transfer* versus synapse weight transfers from memory are preferred since as there are fewer neurons than synapses hence executing fewer external memory fetches. There are several tiles, within each tile are 4 EDRAM banks which has all synapses. Unlike SRAM from TrueNorth [28] EDRAM requires periodic refresh and has higher latency than SRAM. Methods of interleaving are therefore used between the 4EDRAM banks to overcome the destructive read nature of the EDRAMs [135].

In section 5 emerging devices using electron spin to store information are discussed as synaptic emulator candidates, but in SPINDLE [29] a crossbar *spin neuron* is proposed, where a memristive synaptic crossbar is fed to a spintronic comparator incorporated within an enhanced SAR-ADC. In figure 25 the neuron output is generated by sensing the resistance of the MTJ (which represents the comparison of a bias versus the summed input current from the memristor crossbar). Like in [130] this enhanced SAR ADC additionally performs an approximation of the activation function, in this case a hyperbolic tangent (tanh). SPINDLE [29] provides a hierarchical three-tiered architecture composing of spin neuromorphic arrays (SNAs), spin neuromorphic cores (SNCs) and SNC clusters. Within each array (SNA), in figure 25, is a memristive array and spin-neurons, and peripheral circuitry for driving and conv-pooling operations. The SNAs are arranged within cores (SNCs) which also contain local scratchpad memory (to store input features that SNA needs and output features that

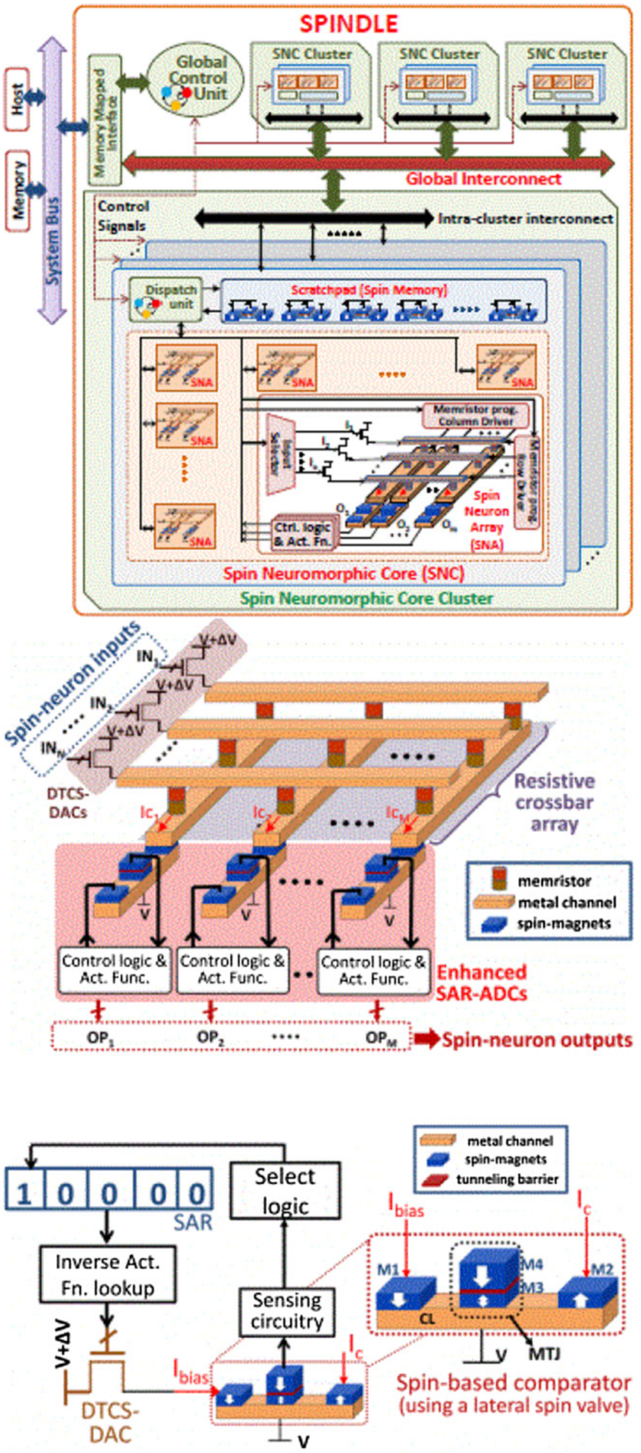


Figure 25. SPINDLE architecture. © 2014 IEEE. Reprinted, with permission from author, from [29].

are generated) and dispatch block to transfer the input features (thus if they share input feature they are inside a core). Cores are then grouped into clusters where each cluster has global interconnect to a shared memory and global control unit. There is a two level memory hierarchy: on-chip distributed scratchpad memories local to SNCs, and off-chip shared memory.

The routing fabric needed for data movement across the memristor crossbar arrays and its communication and synchronization with the CPU using a centralized mesh of the crossbar arrays is conceptualized in RENO [47]. Each group of four (64 × 64) arrays are connected to a group router which is in turn connected to a central router. A routing management solution is proposed for MLP or AAM (auto-associated memory) architectures and how the looping fields are created for the latter in order to determine the destination router address. A switched op-amp based sample and hold circuitry is discussed to buffer the analog signal across the network

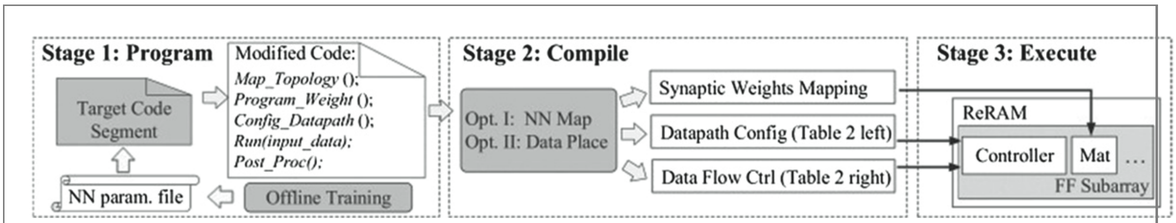


Figure 26. PRIME architecture software perspective: source code program, compile execution. © 2016 IEEE. Reprinted, with permission from author, from [131].

through a multiplexer. In-line calibration is also provided to monitor resistance shift of the memristor arrays and a means to restore them.

Buffering analog signals between the various neural network layers is reduced in ISAAC [46] using pipelining at the expense of more power consumption as all layers are simultaneously active. Using a VGG1 [16] implementation architecture a 16× throughput improvement over a non-pipelined ISAAC was obtained. This computational efficiency also illustrates superior throughput (479–1707GOPS/s × (mm^2) over DaDianNao [135] (63–344GOPS/s × (mm^2). To reduce ADC size, an encoding scheme [46] is introduced where if the sum of products gives an MSB of '1' due to large synaptic weights then the sum of products is flipped so that the MSB is zero thus reducing ADC resolution (a means to flag if a column is in its original or flipped form is also stored). This type of encoding improves ADC efficiency and cell density.

An architecture for memory cells to be switched on demand for neural network computations or storage boosts performance and energy efficiencies in PRIME [45, 131], a dynamically morphable processing-in-memory architecture. Also provided is a software/hardware interface that allows for APIs enabling developer mapping of the neural networks to the ReRAM subarrays, program weights and configure data paths (figure 26). The compile phase optimizes both code and mapping of the neural networks to the sub-arrays to realize small to large scale neural networks specifications (that might require interconnection between banks). Since the PRIME architecture supports both MLP and CNN, a means of pooling layer is discussed with a favoring to the ease of mean-pooling (over max pooling) offered by simply reprogramming a ReRAM subarray ($1/n$) to achieve the desired n:1 mean pooling ratio.

Support for LSTMs is provided in training accelerator PUMA [44] and like [61] offers its own special instruction set architecture and compiler. The bit slicing technique from inference only ISAAC [46] is enhanced in PRIME [61] to support its training implementation where the needed high precision of ReRAM outer product accumulation is accommodated while reducing overhead on ADCs and DACs. The operation is split into time slices covered over several crossbars of the same layer to achieve a 32 bit matrix value. With this implementation introduces the concept of heterogeneous weight slicing where allowing higher precision to the more frequently updated slices is accommodated to reduce device saturation likelihood.

A means of handling the carries from the operations and the frequency of propagating such is also implemented in [61] (carry resolution step). Variants of SGD are also discussed [61] which depending on batch size require replicated copies of a crossbar to prevent structural hazards and avoid additional usage of shared memory.

Similarly described in [5] each DNN model copy is processing and training the same DNN model in parallel but each is observing different training portions of the training database. The weights will tend to diverge for each copy of the DNN model as each is reacting to their own particular/unique training set sequence. Coordination of the various processing nodes, updating a master copy based on the feedback, is needed with an overseer engine to provide DNN training speedups.

Fast programming strategies such as the introduction of coarser control steps by using longer pulses for fast resistance change and adjusting to shorter pulses for finer control [52] can be used. Other means are further investigated in [56] for reducing write latency by comparing the current state of the synaptic cell with the target state to determine if is faster to reach the target by either resetting to start programming from the ReRAM R_{HRS} or issuing a set to start programming from the ReRAM R_{LRS} state hence needing fewer programming iterations to meet the target resistance shown in figure 27. This method is at the expense of retention programs and a more reliable (albeit slower) means of controlling the strengths of the CFs by favoring a rupture of CFs to hit the target value is also proposed [56] shown figure 28. Both these methods are advantageous depending on the application; FPS for *in situ* training where retention requirements are less stringent need reduced latency writes and the more reliable, slower, form of programming for inference application where device retention requirements are needed in figure 14.

Architectures will also require continuous monitoring of synaptic weights so that they can be reset when near the danger zone of saturation to prevent network freeze. Such a method is proposed [129] where training

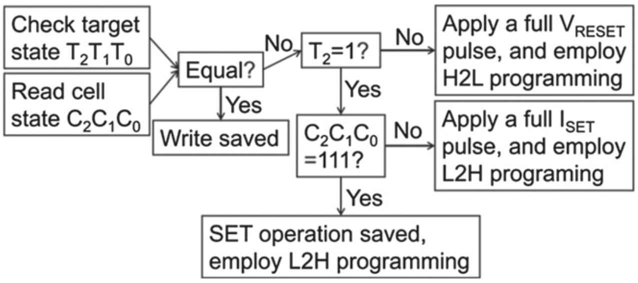

Check target state $T_2T_1T_0$
Read cell state $C_2C_1C_0$
Equal?
No
Yes
Write saved
T_2=1?
No
Yes
Apply a full V_{RESET} pulse, and employ H2L programming
$C_2C_1C_0$ =111?
No
Yes
Apply a full I_{SET} pulse, and employ L2H programing
SET operation saved, employ L2H programming

Figure 27. Fast programming scheme flowchart. © 2013 IEEE. Reprinted, with permission from author, from [56].

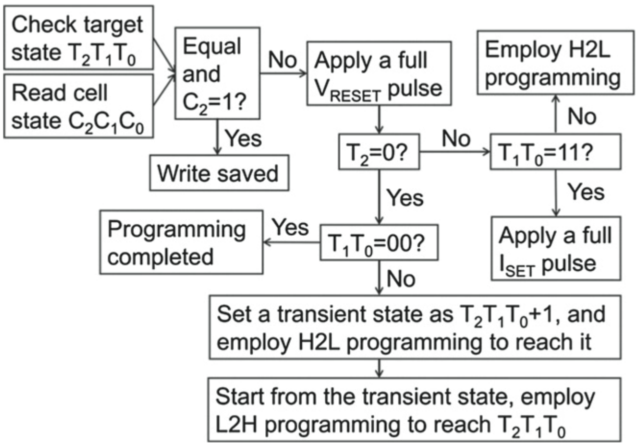

Check target state $T_2T_1T_0$
Read cell state $C_2C_1C_0$
Equal and C_2=1?
No
Yes
Write saved
Apply a full V_{RESET} pulse
Employ H2L programming
T_2=0?
No
Yes
T_1T_0=11?
No
Yes
Apply a full I_{SET} pulse
Programming completed
Yes
T_1T_0=00?
No
Set a transient state as $T_2T_1T_0$+1, and employ H2L programming to reach it
Start from the transient state, employ L2H programming to reach $T_2T_1T_0$

Figure 28. Reliable programming scheme flowchart. © 2013 IEEE. Reprinted, with permission from author, from [56].

is paused, conductance measured, to indicate which conductance of the synaptic pair requires reset and a reset is issued followed by an incremental partial-set to restore/preserve the original conductance difference (albeit far away from the saturation zone). An additional verify step may be needed in highly variable devices adding more write latency. Synapse suppression [51] is another means to 'remove' devices that show dither with frequent updates. Of note is the network configuration usage of devices may cause more 'wear' over another configuration, for example in convolutional nets, set and reset cycles on devices is three orders of magnitude larger than FC nets hence faster device degradation [3]. In the case of [61] the bit-slicing techniques causes more updates in central slices versus edge slices.

This section by no means encompasses all the architectural considerations for NVM based accelerators but provides some insight as to the complexities and multidisciplinary approaches required to make these analog based accelerator architectures viable.

8. Summary

Analog-based accelerators can only be adopted if they provide significant advantages over current processing techniques bench-marked across similar data sets and models. At this stage, while providing some compelling evidence to reduced memory fetches through processing directly in the memory crossbar and increased throughput and parallelism by condensing the number of operations—there are still more questions to be answered regarding handling the non-idealities and variations in the circuitry and devices. In this paper we reviewed various synaptic emulator NVM candidates for in memory computing and the device development required to meet the proposed solution-specific ideal requirements of a synaptic emulator. In consideration is a hybrid of these qualities in order to approach the ideal emulator specifications either within the same layer or across different layers and is an open area for research. The paper also reviews the crossbar sizing limitations, effective line resistance and sneak paths and mitigation of these effects to allow for high density growth. Constant regeneration of the analog signal is needed to propagate the signal through the network through the use of supporting data converter circuitry that in themselves provide bottlenecks requiring precision control compromises. The various architectures so far seem to form a consensus around SAR ADCs, and use of pipelining though several hierarchies of crossbars and sprinkled co-located memories to allow for re-configurability and communication between crossbar nodes and off-chip through global bus networks. The architectural perspective of current concept analog based accelerators propose techniques to overcome some of the challenges for device variation, retention, endurance, and circuit non-idealities while still maintaining comparative quality classification accuracies (to their digital counterparts) as well as learnings for further research. They are not at the stage yet to accommodate the larger elaborate models and data sets in use today to benchmark against current commercial accelerators.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. CISE-1910881.

Data availability statement

No new data were created or analysed in this study.

ORCID iDs

Mirembe Musisi-Nkambwe https://orcid.org/0000-0002-1129-3924
Sahra Afshari https://orcid.org/0000-0002-2718-1812
Hugh Barnaby https://orcid.org/0000-0002-8136-1849
Michael Kozicki https://orcid.org/0000-0003-1281-6827
Ivan Sanchez Esqueda https://orcid.org/0000-0001-6530-8602

References

[1] Sebastian A, Le Gallo M, Khaddam-Aljameh R and Eleftheriou E 2020 Memory devices and applications for in-memory computing *Nat. Nanotechnol.* **15** 529

[2] Haensch W, Gokmen T and Puri R 2019 The next generation of deep learning hardware: analog computing *Proc. IEEE* **107** 108–22

[3] Tsai H, Ambrogio S, Narayanan P, Shelby R M and Burr G W 2018 Recent progress in analog memory-based accelerators for deep learning *J. Phys. D: Appl. Phys.* **51** 283001

[4] Sze V, Chen Y-H, Yang T-J and Emer J S 2017 Efficient processing of deep neural networks: a tutorial and survey *Proc. IEEE* **105** 2295–329

[5] Narayanan P, Fumarola A, Sanches L L, Hosokawa K, Lewis S C, Shelby R M and Burr G W 2017 Toward on-chip acceleration of the backpropagation algorithm using nonvolatile memory *IBM J. Res. Dev.* **61** 11

[6] Fumarola A, Narayanan P, Sanches L L, Sidler S, Jang J, Moon K, Shelby R M, Hwang H and Burr G W 2016 Accelerating machine learning with non-volatile memory: exploring device and circuit tradeoffs *2016 IEEE Int. Conf. on Rebooting Computing (ICRC)* pp1–8

[7] Choi S, Sheridan P and Lu W D 2015 Data clustering using memristor networks *Sci. Rep.* **5** 10492

[8] Sheridan P M, Du C and Lu W D 2016 Feature extraction using memristor networks *IEEE Trans. Neural Netw. Learn. Syst.* **27** 2327–36

[9] Pfeiffer M and Pfeil T 2018 Deep learning with spiking neurons: opportunities and challenges *Front. Neurosci.* **12** 774

[10] Schemmel J, Grubl A, Meier K and Mueller E 2006 Implementing synaptic plasticity in a vlsi spiking neural network model *The 2006 IEEE Int. Joint Conf. on Neural Network Proceedings* pp1–6

[11] ML Commons MLPerf fair and useful benchmarks for measuring training and inference performance of ml hardware, software, and services https://mlperf.org/ (accessed: 30 September 2020)

[12] Mattson P *et al* 2020 *Mlperf Training Benchmark*

[13] Reddi V J *et al* 2019 *Mlperf Inference Benchmark*

[14] Lecun Y, Bottou L, Bengio Y and Haffner P 1998 Gradient-based learning applied to document recognition *Proc. IEEE* **86** 2278–324

[15] Krizhevsky A, Sutskever I and Hinton G E 2012 ImageNet classification with deep convolutional neural networks *Commun. ACM* **60** 84–90

[16] Simonyan K and Zisserman A 2015 Very deep convolutional networks for large-scale image recognition *3rd Int. Conf. on Learning Representations, ICLR 2015, Conf. Track Proceedings* (San Diego, CA, USA May 7–9, 2015) ed Y Bengio and Y LeCun

[17] Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V and Rabinovich A 2015 Going deeper with convolutions *2015 IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)* pp1–9

[18] He K, Zhang X, Ren S and Sun J 2015 Deep residual learning for image recognition (arXiv:1512.03385)

[19] Devlin J, Chang M, Lee K and Toutanova K 2018 BERT: pre-training of deep bidirectional transformers for language understanding (arXiv:1810.04805)

[20] Wikipedia Contributors 2021 https://en.wikipedia.org/wiki/List_of_datasets_for_machine-learning_research List of datasets for machine-learning research—Wikipedia, the free encyclopedia (accessed 11 April 2021)

[21] Zue V, Seneff S and Glass J 1990 Speech database development at mit: timit and beyond *Speech Commun.* **9** 351–6

[22] Georghiades A Center for Computational Vision and Control at Yale University 1997 (Accessed 7 January 2021) Yale face database http://cvc.cs.yale.edu/cvc/projects/yalefaces/yalefaces.html

[23] LeCun Y 1998 MNIST database http://yann.lecun.com/exdb/mnist/ (accessed 30 September 2020)

[24] Krizhevsky A The cifar-10 dataset https://cs.toronto.edu/\simkriz/cifar.html

[25] Geiger A, Lenz P and Urtasun R 2012 Are we ready for autonomous driving? The kitti vision benchmark suite *2012 IEEE Conf. on Computer Vision and Pattern Recognition* pp 3354–61

[26] Schmidhuber J 2015 Deep learning in neural networks: an overview *Neural Netw.* **61** 85–117

[27] Holler M, Tam C and Benson R G 1989 An electrically trainable artificial neural network (ETANN) with 10240 'floating gate' synapses *Int. 1989 Joint Conf. on Neural Networks* vol 2 pp 191–6

[28] Merolla P A *et al* 2014 A million spiking-neuron integrated circuit with a scalable communication network and interface *Science* **345** 668–73

[29] Ramasubramanian S G, Venkatesan R, Sharad M, Roy K and Raghunathan A 2014 Spindle: spintronic deep learning engine for large-scale neuromorphic computing *2014 IEEE/ACM Int. Symp. on Low Power Electronics and Design (ISLPED)* pp 15–20

[30] Jouppi N P *et al* 2017 In-datacenter performance analysis of a tensor processing unit *SIGARCH Comput. Architect. News* **45** 1–12

[31] Farabet C, Martini B, Corda B, Akselrod P, Culurciello E and LeCun Y 2011 Neuflow: a runtime reconfigurable dataflow processor for vision *CVPR 2011 Workshops* pp109–16

[32] Chen T, Du Z, Sun N, Wang J, Wu C, Chen Y and Temam O 2014 DianNao *Proc. of the 19th Int. Conf. on Architectural Support for Programming Languages and Operating Systems, Ser. ASPLOS '14* **vol 49** (New YorkAssociation for Computing Machinery) pp269–84

[33] Kumar N 2020 (Accessed 3 September 2020) https://cloud.google.com/blog/products/ai-machine-learning/google-breaks-ai-performance-records-in-mlperf-with-worlds-fastest-training-supercomputer Google breaks AI performance records in MLPerf with world's fastest training supercomputer https://cloud.google.com/tpu

[34] NVIDIA [WHITE PAPER] 2020 https://images.nvidia.com/aem-dam/en-zz/Solutions/data-center/nvidia-ampere-architecture-whitepaper.pdf (accessed 13 March 2021)

[35] Merenda M, Porcaro C and Iero D 2020 Edge machine learning for AI-enabled IOT devices: a review *Sensors* **20** 2533

[36] Injong R 2018 https://www.blog.google/products/google-cloud/bringing-intelligence-to-the-edge-with-cloud-iot/ Bringing intelligence to the edge with Cloud IoT (accessed 30 September 2020)

[37] Li C *et al* 2019 Long short-term memory networks in memristor crossbar arrays *Nat. Mach. Intell.* **1** 49–57

[38] Shaikh F 2018 An introduction to pytorch—a simple yet powerful deep learning library http://analyticsvidhya.com/blog/2018/02/pytorch-tutorial/ (accessed 13 March 2021)

[39] Abadi M *et al* 2016 Tensorflow: large-scale machine learning on heterogeneous distributed systems (arXiv:1603.04467)

[40] Sze V, Chen Y-H, Emer J, Suleiman A and Zhang Z 2017 Hardware for machine learning: challenges and opportunities *2017 IEEE Custom Integrated Circuits Conf. (CICC)*

[41] Zoabi Y, Deri-Rozov S and Shomron N 2021 Machine learning-based prediction of COVID-19 diagnosis based on symptoms *npj Digit. Med.* **4** 3

[42] Medina E and Dagan E 2020 Habana labs purpose-built AI inference and training processor architectures: scaling AI training systems using standard ethernet with gaudi processor *IEEE Micro* **40** 17–24

[43] Nikonov D E and Young I A 2019 Benchmarking delay and energy of neural inference circuits *IEEE J. Explor. Solid-State Comput. Devices Circuits* **5** 75–84

[44] Ankit A *et al* 2019 PUMA: a programmable ultra-efficient memristor-based accelerator for machine learning inference (arXiv:1901.10351)

[45] Chi P, Li S, Xu C, Zhang T, Zhao J, Liu Y, Wang Y and Xie Y 2016 Prime *SIGARCH Comput. Architect. News* **44** 27–39

[46] Shafiee A, Nag A, Muralimanohar N, Balasubramonian R, Strachan J P, Hu M, Williams R S and Srikumar V 2016 Isaac *2016 ACM/IEEE 43rd Annual Int. Symp. on Computer Architecture (ISCA)* **vol 44** pp14–26

[47] Liu X *et al* 2015 Reno: a high-efficient reconfigurable neuromorphic computing accelerator design *2015 52nd ACM/EDAC/IEEE Design Automation Conf. (DAC)* pp1–6

[48] Li C *et al* 2018 Analogue signal and image processing with large memristor crossbars *Nat. Electron.* **1** 52–9

[49] Merolla P, Arthur J, Akopyan F, Imam N, Manohar R and Modha D S 2011 A digital neurosynaptic core using embedded crossbar memory with 45 pJ per spike in 45 nm *2011 IEEE Custom Integrated Circuits Conf. (CICC)* pp1–4

[50] Seo J *et al* 2011 A 45 nm CMOS neuromorphic chip with a scalable architecture for learning in networks of spiking neurons *2011 IEEE Custom Integrated Circuits Conf. (CICC)* pp 1–4

[51] Narayanan P, Sanches L L, Fumarola A, Shelby R M, Ambrogio S, Jang J, Hwang H, Leblebici Y and Burr G W 2017 Reducing circuit design complexity for neuromorphic machine learning systems based on non-volatile memory arrays *2017 IEEE Int. Symp. on Circuits and Systems (ISCAS)* pp1–4

[52] Chen W, Fang R, Balaban M B, Yu W, Gonzalez-Velo Y, Barnaby H J and Kozicki M N 2016 A CMOS-compatible electronic synapse device based on Cu/SiO$_2$/W programmable metallization cells *Nanotechnology* **27** 255202

[53] Li B, Gu P, Shan Y, Wang Y, Chen Y and Yang H 2015 RRAM-based analog approximate computing *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **34** 1905–17

[54] Sanchez Esqueda I *et al* 2018 Aligned carbon nanotube synaptic transistors for large-scale neuromorphic computing *ACS Nano* **12** 7352–61

[55] Suri M, Bichler O, Querlioz D, Cueto O, Perniola L, Sousa V, Vuillaume D, Gamrat C and DeSalvo B 2011 Phase change memory as synapse for ultra-dense neuromorphic systems: application to complex visual pattern extraction *2011 Int. Electron Devices Meeting* pp14.4.–4

[56] Xu C, Niu D, Muralimanohar N, Jouppi N P and Yuan X 2013 Understanding the trade-offs in multi-level cell reram memory design *2013 50th ACM/EDAC/IEEE Design Automation Conf. (DAC)* pp1–6

[57] Jackson B L *et al* 2013 Nanoscale electronic synapses using phase change devices *J. Emerg. Technol. Comput. Syst.* **9** 1

[58] Chekol S A, Song J, Park J, Yoo J, Lim S and Hwang H 2020 Selector devices for emerging memories *Memristive Devices for Brain-Inspired Computing* (*Woodhead Publishing Series in Electronic and Optical Materials*) ed S Spiga, A Sebastian, D Querlioz and B Rajendran (United Kingdom: Woodhead Publishing) pp 135–64

[59] Agarwal S, Quach T-T, Parekh O, Hsia A H, DeBenedictis E P, James C D, Marinella M J and Aimone J B 2016 Energy scaling advantages of resistive memory crossbar based computation and its application to sparse coding *Front. Neurosci.* **9** 484

[60] Kadetotad D *et al* 2015 Parallel architecture with resistive crosspoint array for dictionary learning acceleration *IEEE J. Emerg. Sel. Top. Circuits Syst.* **5** 194–204

[61] Ankit A *et al* 2019 Panther: a programmable architecture for neural network training harnessing energy-efficient ReRAM (arXiv:1912.11516)

[62] Linares-Barranco B, Serrano-Gotarredona T, Camuñas-Mesa L, Perez-Carrasco J, Zamarreño-Ramos C and Masquelier T 2011 On spike-timing-dependent-plasticity, memristive devices, and building a self-learning visual cortex *Front. Neurosci.* **5** 26

[63] Chakraborty I, Ali M, Ankit A, Jain S, Roy S, Sridharan S, Agrawal A, Raghunathan A and Roy K 2020 Resistive crossbars as approximate hardware building blocks for machine learning: opportunities and challenges *Proc. IEEE* **108** 2276

[64] Yao P, Wu H, Gao B, Tang J, Zhang Q, Zhang W, Yang J J and Qian H 2020 Fully hardware-implemented memristor convolutional neural network *Nature* **577** 641–6

[65] Yakopcic C and Taha T M 2016 Model for maximum crossbar size based on input driver impedance *Electron. Lett.* **52** 25–7

[66] Amer S and Rose G S 2019 A multi-driver write scheme for reliable and energy efficient 1S1R ReRAM crossbar arrays *20th Int. Symp. on Quality Electronic Design (ISQED)* pp 64–9

[67] Gu P, Li B, Tang T, Yu S, Cao Y, Wang Y and Yang H 2015 Technological exploration of RRAM crossbar array for matrix-vector multiplication *The 20th Asia and South Pacific Design Automation Conf.* pp 106–11

[68] Xu C, Niu D, Muralimanohar N, Balasubramonian R, Zhang T, Yu S and Xie Y 2015 Overcoming the challenges of crossbar resistive memory architectures *2015 IEEE 21st Int. Symp. on High Performance Computer Architecture (HPCA)* pp 476–88

[69] Gül F 2019 Addressing the sneak-path problem in crossbar RRAM devices using memristor-based one Schottky diode-one resistor array *Results Phys.* **12** 1091–6

[70] Gonzalez-Velo Y, Barnaby H J and Kozicki M N 2017 Review of radiation effects on ReRAM devices and technology *Semicond. Sci. Technol.* **32** 083002

[71] Yu S, Chen P, Cao Y, Xia L, Wang Y and Wu H 2015 Scaling-up resistive synaptic arrays for neuro-inspired architecture: challenges and prospects *2015 IEEE Int. Electron Devices Meeting (IEDM)* 17.3.pp 1–4

[72] Krestinskaya O, Salama K N and James A P 2019 Learning in memristive neural network architectures using analog backpropagation circuits *IEEE Trans. Circuits Syst.* I **66** 719–32

[73] Park J, Hadamek T, Posadas A, Cha E, Demkov A and Hwang H 2017 Multi-layered $\mathrm{NiO}_y/\mathrm{NbO}_x/\mathrm{NiO}_y$ fast drift-free threshold switch with high $I_{\mathrm{on}}/I_{\mathrm{off}}$ ratio for selector application *Sci. Rep.* **7** 4068

[74] Yoo J, Koo Y, Chekol S, Park J, Song J and Hwang H 2018 Te-based binary OTS selectors with excellent selectivity ($>10^5$), endurance ($>10^8$) and thermal stability ($>450\ ^\circ$C) *2018 IEEE Symp. on VLSI Technology* pp 207–8

[75] Jain S, Sengupta A, Roy K and Raghunathan A 2020 RxNN: a framework for evaluating deep neural networks on resistive crossbars *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **40** 326

[76] Hermiz J, Chang T, Du C and Lu W 2013 Interference and memory capacity effects in memristive systems *Appl. Phys. Lett.* **102** 083106

[77] Berdan R, Vasilaki E, Khiat A, Indiveri G, Serb A and Prodromakis T 2015 Emulating short-term synaptic dynamics with memristive devices *Sci. Rep.* **6** 18639

[78] Kim H, Nili H, Mahmoodi M and Strukov D 2019 4k-memristor analog-grade passive crossbar circuit (arXiv:1906.12045v1)

[79] Murali G, Sun X, Yu S and Lim S K 2021 Heterogeneous mixed-signal monolithic 3D in-memory computing using resistive ram *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **29** 386–96

[80] Gonugondla S K, Kang M and Shanbhag N R 2018 A variation-tolerant in-memory machine learning classifier via on-chip training *IEEE J. Solid-State Circuits* **53** 3163–73

[81] Yin S *et al* 2019 Monolithically integrated RRAM- and CMOS-based in-memory computing optimizations for efficient deep learning *IEEE Micro* **39** 54–63

[82] Ramkumar K, Venkataraman P and Geha S 2019 [WHITE PAPER] Cypress SONOS technology https://cypress.com/file/123341/download (accessed 30 September 2020)

[83] Jain P *et al* 2019 13.2 a 3.6 mb 10.1 mb/mm^2 embedded non-volatile ReRAM macro in 22 nm FINFET technology with adaptive forming/set/reset schemes yielding down to 0.5 V with sensing time of 5 ns at 0.7 V *2019 IEEE Int. Solid-State Circuits Conference (ISSCC)* pp 212–4

[84] Chou C, Lin Z, Tseng P, Li C, Chang C, Chen W, Chih Y and Chang T J 2018 An *n*40 256k × 44 embedded RRAM macro with SL-precharge SA and low-voltage current limiter to improve read and write performance *2018 IEEE Int. Solid-State Circuits Conference (ISSCC)* pp 478–80

[85] Swaroop B, West W C, Martinez G, Kozicki M N and Akers L A 1998 Programmable current mode hebbian learning neural network using programmable metallization cell *1998 IEEE Int. Symp. on Circuits and Systems (ISCAS)* vol 3 pp 33–6

[86] Edwards A H, Barnaby H J, Campbell K A, Kozicki M N, Liu W and Marinella M J 2015 Reconfigurable memristive device technologies *Proc. IEEE* **103** 1004–33

[87] Kozicki M N and Barnaby H J 2016 Conductive bridging random access memory-materials, devices and applications *Semicond. Sci. Technol.* **31** 113001

[88] Wong H-S P, Lee H-Y, Yu S, Chen Y-S, Wu Y, Chen P-S, Lee B, Chen F T and Tsai M-J 2012 Metal-oxide RRAM *Proc. IEEE* **100** 1951–70

[89] Sanchez Esqueda I, Zhao H and Wang H 2018 Efficient learning and crossbar operations with atomically-thin 2D material compound synapses *J. Appl. Phys.* **124** 152133

[90] Eryilmaz S B, Kuzum D, Yu S and Wong H P 2015 Device and system level design considerations for analog-non-volatile-memory based neuromorphic architectures *2015 IEEE Int. Electron Devices Meeting (IEDM)* 4.1.pp 1–4

[91] Merrikh-Bayat F, Prezioso M, Chakrabarti B, Kataeva I and Strukov D 2018 Implementation of multilayer perceptron network with highly uniform passive memristive crossbar circuits *Nat. Commun.* **9** 2331

[92] Guan X, Yu S and Wong H-S P 2012 A spice compact model of metal oxide resistive switching memory with variations *IEEE Electron Device Lett.* **33** 1405–7

[93] Li C *et al* 2018 Large memristor crossbars for analog computing *2018 IEEE Int. Symp. on Circuits and Systems (ISCAS)* pp 1–4

[94] Chen A and Lin M 2011 Variability of resistive switching memories and its impact on crossbar array performance *2011 Int. Reliability Physics Symp.* MY.7.1–4

[95] Chang T, Jo S-H and Lu W 2011 Short-term memory to long-term memory transition in a nanoscale memristor *ACS Nano* **5** 7669–76

[96] Li C *et al* 2018 Efficient and self-adaptive *in situ* learning in multilayer memristor neural networks *Nat. Commun.* **9** 2385

[97] Trentzsch M *et al* 2016 A 28 nm HKMG super low power embedded NVM technology based on ferroelectric fets *2016 IEEE Int. Electron Devices Meeting (IEDM)* 11.5.pp 1–4

[98] Kim T and Lee S 2020 Evolution of phase-change memory for the storage-class memory and beyond *IEEE Trans. Electron Devices* **67** 1394–406

[99] Gallo M, Krebs D, Zipoli F, Salinga M and Sebastian A 2018 Collective structural relaxation in phase-change memory devices *Adv. Electron. Mater.* **4** 1700627

[100] Jasemi M, Hessabi S and Bagherzadeh N 2020 Reliable and energy efficient MLC STT-RAM buffer for CNN accelerators *Comput. Electron. Eng.* **86** 106698

[101] Mulaosmanovic H, Ocker J, Müller S, Noack M, Müller J, Polakowski P, Mikolajick T and Slesazeck S 2017 Novel ferroelectric FET based synapse for neuromorphic systems *2017 Symp. on VLSI Technology* Tpp 176–7

[102] Sharma A A *et al* 2020 High speed memory operation in channel-last, back-gated ferroelectric transistors *2020 IEEE Int. Electron Devices Meeting (IEDM)* 18.5.pp 1–4

[103] Soliman T *et al* 2020 Ultra-low power flexible precision FEFET based analog in-memory computing *2020 IEEE Int. Electron Devices Meeting (IEDM)* 29.2.pp 1–4

[104] Zeng B, Liao M, Peng Q, Xiao W, Liao J, Zheng S and Zhou Y 2019 2 bit/cell operation of $Hf_{0.5}Zr_{0.5}O_2$ based FEFET memory devices for NAND applications *IEEE J. Electron Devices Soc.* **7** 551–6

[105] Yao P *et al* 2017 Face classification using electronic synapses *Nat. Commun.* **8** 15199

[106] Peng X, Huang S, Luo Y, Sun X and Yu S 2019 DNN + neurosim: an end-to-end benchmarking framework for compute-in-memory accelerators with versatile device technologies *2019 IEEE Int. Electron Devices Meeting (IEDM)* 32.5.pp 1–4

[107] Agarwal S, Plimpton S, Richter I, Hsia A and Hughart D 2018 Crosssim http://cross-sim.sandia.gov

[108] Prezioso M, Merrikh-Bayat F, Hoskins B D, Adam G C, Likharev K K and Strukov D B 2015 Training and operation of an integrated neuromorphic network based on metal-oxide memristors *Nature* **521** 61–4

[109] Bao G, Zhang Y and Zeng Z 2020 Memory analysis for memristors and memristive recurrent neural networks *IEEE/CAA J. Autom. Sinica* **7** 96–105

[110] Jain S and Raghunathan A 2020 CxDNN *ACM Trans. Embed. Comput. Syst.* **18** 1

[111] Han S, Mao H and Dally W 2016 Deep compression: compressing deep neural network with pruning, trained quantization and huffman coding (arXiv:1510.00149v5)

[112] Wang Y, Wen W, Song L and Li H H 2017 Classification accuracy improvement for neuromorphic computing systems with one-level precision synapses *2017 22nd Asia and South Pacific Design Automation Conf. (ASP-DAC)* pp 776–81

[113] Mohanty A, Du X, Chen P, Seo J, Yu S and Cao Y 2017 Random sparse adaptation for accurate inference with inaccurate multi-level RRAM arrays *2017 IEEE Int. Electron Devices Meeting (IEDM)* 6.3.pp 1–4

[114] Ambrogio S *et al* 2018 Equivalent-accuracy accelerated neural-network training using analogue memory *Nature* **558** 60

[115] Cai F, Correll J, Lee S H, Lim Y, Bothra V, Zhang Z, Flynn M and Lu W 2019 A fully integrated reprogrammable memristor-CMOS system for efficient multiply-accumulate operations *Nat. Electron.* **2** 290

[116] Sun X, Yin S, Peng X, Liu R, Seo J and Yu S 2018 XNOR-RRAM: a scalable and parallel resistive synaptic architecture for binary neural networks *2018 Design, Automation Test in Europe Conf. Exhibition (DATE)* pp 1423–8

[117] Jiang H, Li W, Huang S, Cosemans S, Catthoor F and Yu S 2021 Analog-to-digital converter design exploration for compute-in-memory accelerators *IEEE Des. Test* **0** 1-1

[118] Li J *et al* 2011 A novel reconfigurable sensing scheme for variable level storage in phase change memory *2011 3rd IEEE Int. Memory Workshop (IMW)* pp 1–4

[119] Chan K L *et al* 2016 A 32.75 gb s^{-1} voltage mode transmitter with 3-tap FFE in 16 nm CMOS *2016 IEEE Asian Solid-State Circuits Conf. (A-SSCC)* pp 233–6

[120] Wilson H and Haycock M 2001 A six-port 30 gb s^{-1} nonblocking router component using point-to-point simultaneous bidirectional signaling for high-bandwidth interconnects *IEEE J. Solid-State Circuits* **36** 1954–63

[121] Xue C-X *et al* 2020 A CMOS-integrated compute-in-memory macro based on resistive random-access memory for AI edge devices *Nat. Electron.* **4** 81–90

[122] Xue C X *et al* 2021 16.1 a 22 nm 4 mb 8b-precision ReRAM computing-in-memory macro with 11.91 to 195.7 tops/w for tiny AI edge devices *2021 IEEE Int. Solid-State Circuits Conf. (ISSCC)* vol 64 pp 245–7

[123] Sinha M and Burleson W 2001 Current-sensing for crossbars *Proc. 14th Annual IEEE Int. ASIC/SOC Conf. (IEEE Cat. No. 01TH8558)* pp 25–9

[124] Chang M-F, Lee A, Chen P-C, Lin C J, King Y-C, Sheu S-S and Ku T-K 2015 Challenges and circuit techniques for energy-efficient on-chip nonvolatile memory using memristive devices *IEEE J. Emerg. Sel. Top. Circuits Syst.* **5** 183–93

[125] Yu S, Sun X, Peng X and Huang S 2020 Compute-in-memory with emerging nonvolatile-memories: challenges and prospects *2020 IEEE Custom Integrated Circuits Conf. (CICC)* pp 1–4

[126] Uddin M and Rose G 2018 A practical sense amplifier design for memristive crossbar circuits (puf) *2018 31st IEEE Int. System-on-Chip Conf. (SOCC)* vol 9 pp 209–14

[127] Mohammad B, Dadabhoy P, Lin K and Bassett P 2012 Comparative study of current mode and voltage mode sense amplifier used for 28 nm SRAM *2012 24th Int. Conf. on Microelectronics (ICM)* pp 1–6

[128] Gokmen T and Vlasov Y 2016 Acceleration of deep neural network training with resistive cross-point devices: design considerations *Front. Neurosci.* **10** 333

[129] Burr G W *et al* 2014 Experimental demonstration and tolerancing of a large-scale neural network (165 000 synapses), using phase-change memory as the synaptic weight element *2014 IEEE Int. Electron Devices Meeting* 29.5.pp 1–4

[130] Boser B E, Sackinger E, Bromley J, Le Cun Y and Jackel L D 1991 An analog neural network processor with programmable topology *IEEE J. Solid-State Circuits* **26** 2017–25

[131] Chi P, Li S, Xu C, Zhang T, Zhao J, Liu Y, Wang Y and Xie Y 2016 Prime *Proc. of the 43rd Int. Symp. on Computer Architecture, Ser. ISCA'16* vol 44 (IEEE Press) pp 27–39

[132] Zhao Z, Qu L, Wang L, Deng Q, Li N, Kang Z, Guo S and Xu W 2020 A memristor-based spiking neural network with high scalability and learning efficiency *IEEE Trans. Circuits Syst. II* **67** 931

[133] Ankit A, Sengupta A and Roy K 2017 Trannsformer: neural network transformation for memristive crossbar based neuromorphic system design *2017 IEEE/ACM Int. Conf. on Computer-Aided Design (ICCAD)*

[134] Boser B E, Sackinger E, Bromley J, LeCun Y, Howard R E and Jackel L D 1991 An analog neural network processor and its application to high-speed character recognition *IJCNN-91-Seattle Int. Joint Conf. on Neural Networks* vol I pp 415–20

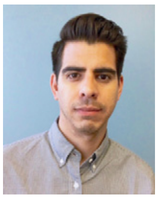
[135] Luo T, Liu S, Li L, Wang Y, Zhang S, Chen T, Xu Z, Temam O and Chen Y 2017 Dadiannao: a neural network supercomputer *IEEE Trans. Comput.* **66** 73–88

Mirembe Musisi-Nkambwe has a B.S and M.S in Electrical Engineering from the Rochester Institute of Technology, NY, USA. She is a Ph.D. candidate in Electrical Engineering in the School of Electrical, Computer, and Energy Engineering at Arizona State University. (ASU), Tempe, AZ, USA. She is a 17 year veteran in the semiconductor industry as a Circuit Design engineer at Intel Corp and previously at Freescale Semiconductor (now NXP Semiconductors) in Chandler, AZ, USA.

Sahra Afshari has a BS in Computer and Electrical Engineering from Azad University, Mashhad, Iran, and MS in Electrical Engineering from Arizona State University (ASU), Tempe, AZ. Currently. She is a Ph.D. candidate in Electrical Engineering in the school of Electrical, Computer, and Energy Engineering in Arizona State University (ASU), Tempe, AZ. She has a few years of experience as an automation design engineer in the power plant industry. Currently, she specializes in the semiconductor domain.

Hugh Barnaby received his Ph.D. from Vanderbilt University in electrical engineering and has been a member of the faculty at Arizona State University since 2004. His primary research focuses on the analysis, modeling, and experimental characterization of extreme environment effects in semiconductor materials, devices and integrated circuits. As part of this research, he also develops design and processing techniques that enable the reliable operation of electronics in these environments. Dr. Barnaby has ongoing research activities in wireless (RF and optical) IC and data converter design, radiation- and reliability-enabled compact modeling, and memristor technologies and neuromorphic applications.

Michael Kozicki, Professor in the School of Electrical, Computer and Energy Engineering, joined Arizona State University in 1985 from the semiconductor industry. His research on the development of novel solid-state materials, processes, and devices has led to over 60 US patents, several dozen international patents, and commercialized products, achievements which led to his election as a Fellow of the National Academy of Inventors in 2015. He was a Fulton Entrepreneurial Professor from 2016 to 2018 and received the 2019 Daniel Jankowski Legacy Award and 2019-2020 Joseph C. Palais Distinguished Faculty Scholar Award. Dr. Kozicki has also served as Director of entrepreneurial programs and Director of the Center for Solid State Electronics Research in the Ira A. Fulton Schools of Engineering.



Ivan Sanchez Esqueda is currently an Assistant Professor of Electrical Engineering at Arizona State University. He received his Ph.D. from Arizona State University in 2011, and then worked as a research scientist at the University of Southern California for 7 years. His current research is focused on low-dimensional materials, nanoelectronics, and the development of new solid-state technologies for computing and memory applications.