IPSJ Transactions on System LSI Design Methodology Vol.12 53–64 (Aug. 2019)

[DOI: 10.2197/ipsjtsldm.12.53]

Invited Paper

Neuromorphic Computing Systems: From CMOS To Emerging Nonvolatile Memory

Chaofei Yang[1,a)] Ximing Qiao[1,b)] Yiran Chen[1,c)]

Received: April 21, 2019

Abstract: The end of Moore's Law and von Neumann bottleneck motivate researchers to seek alternative architectures that can fulfill the increasing demand for computation resources which cannot be easily achieved by traditional computing paradigm. As one important practice, neuromorphic computing systems (NCS) are proposed to mimic biological behaviors of neurons and synapses, and accelerate computation of neural networks. Traditional CMOS-based implementation of NCS, however, are subject to large hardware cost required to precisely replicate the biological properties. In very recent decade, emerging nonvolatile memory (eNVM) was introduced to NCS design due to its high computing efficiency and integration density. Similar to the circuits built on other nanoscale devices, eNVM-based NCS also suffers from many reliability issues. In this paper, we give a short survey about CMOS- and eNVM-based NCS, including their basic implementations and training and inference schemes in various applications. We also discuss the design challenges of these NCS and introduce some techniques that can improve the reliability, precision, scalability, and security of the NCS. At the end, we provide our insights on the design trend and future challenges of the NCS.

Keywords: Neuromorphic computing, emerging nonvolatile memory, eNVM, brain-inspired computing, reliability, robustness, security.

1. Introduction

The modern computer architecture defined by von Neumann in 1945 (a.k.a. von Neumann architecture) separates memory and processing unit, leading to extra data movement cost between these two components during execution. This issue, which is often referred to as "von Neumann bottleneck [1]" or "memory wall", is becoming very severe when Moore's Law is approaching its end: the increasing of memory bandwidth is far behind the up-scaling of on-chip computing capacity. von Neumann architecture has been also proven inefficient to perform cognitive applications, such as neural network and graph-computing, the computation of which heavily replies on topological data movement. Researchers are actively looking for alternative (i.e., non-von Neumann) architectures to fulfill the increasing demand for computation resources that cannot be easily achieved by traditional computing paradigms. Inspired by human brains, many circuits and systems were invented to mimic the biological properties of neurons and synapses, and build a hardware-coded neural network. Such a design was firstly named by Carver Mead in 1990s as neuromorphic computing [2], which is anticipated to be computing efficiently due to its high parallelism and simple realization of basic units.

Although the actual mechanism of a human brain is very complex [3], it can still be simplified as a connecting model of neurons and synapses [2]. Once a neuron receives sufficient excitation, i.e., the neuron is activated, it will transmit its output signal to the following neuron via a synapse. The signal can be amplified by the weight of the synapse during the transmission. Although the function of the neuron or the synapse is simple, a large number of neurons that are connected through many synapses is able to construct a neural network which can perform very complex functions such as image classification, pattern recognition, natural language processing, etc. Traditional CMOS-based implementations of neuromorphic computing system (NCS) use SRAM to store the weights of the synapses and need very complex CMOS circuitry to realize neuron designs (e.g., the integrate-and-fire circuit) [4], [5], [6], [7], [8]. Substantial power and computing improvements have been accomplished by CMOS-based NCS compared with traditional von Neumann architecture when running neuromorphic computing algorithms. However, CMOS-based NCS generally suffer from two major design challenges: First, the complex implementations of neurons and synapses in CMOS-based NCS prevent the power efficiency of the system (~25 pJ/Op) from scaling down to the level close to a human brain (~10 fJ/Op); Second, CMOS-based NCS generally cannot realize very dense connectivity between the neurons, leading to severe constraint on the scale of the network that can be implemented, i.e., 1 million neurons of a CMOS-based NCS vs. 100 billion neurons of a human brain [9]. In summary, the capacity of CMOS-based NCS is still far away from the level that a human brain can achieve. In the recent decade, hence, researchers start to seek other alternative approaches to implement NCS.

The design based on emerging nonvolatile memory (eNVM)

[1] Duke University, Durham, NC, USA
[a)] chaofei.yang@duke.edu
[b)] ximing.qiao@duke.edu
[c)] yiran.chen@duke.edu

is a popular practice of recent NCS designs. Different from DRAM and SRAM, eNVM is a special type of memory that can retain data even when the power is turned off. Examples of eNVM include phase change memory (PCM) [10], spin-transfer torque random-access memory (STT-RAM) [11], and resistive RAM (RRAM) [12], etc. These technologies have been proven able to scale the size of the memory devices down to several nanometers. A crossbar structure was also proposed to integrate the devices in a very dense way so that every input is connected to all the outputs [13], offering very high operational parallelism and power efficiency. Similar to the circuits built on other nanoscale devices, eNVM-based NCS also suffers from many reliability issues. Therefore, many architecture- and circuit-level solutions were proposed to accelerate execution of the eNVM-based NCS [14], [15], [16], and enhance its robustness [17], [18], [19] and security [20], [21], [22].

In this paper, we give a short survey about CMOS- and eNVM-based NCS, including their basic implementations and training and inference schemes in various applications. We also discuss the design challenges of these NCS and introduce some techniques that can improve the reliability, precision, scalability, and security of the NCS. In the end, we provide our insights on the design trend and future challenges of the NCS. Our paper is organized as follows. Section 2 briefly introduces eNVMs and two variations of neural networks; Section 3 presents different CMOS-based NCS and their limitations. Section 4 introduces eNVM-based neuromorphic designs and compares them with the CMOS-based counterparts. Section 5 summarizes some solutions to address the design challenges of eNVM-based NCS. Sections 6 and 7 share our insights on the future of NCS design and concludes our paper, respectively.

2. Preliminary

eNVM usually denotes the memory technologies that use resistance to store information. In this section, we briefly introduce three types of eNVMs: PCM, STT-RAM, and RRAM, including their working mechanisms and basic implementations. We then introduce the basics of deep neural network (DNN) and spiking neural network (SNN), including their pros and cons for neuromorphic computing.

2.1 Emerging Nonvolatile Memories (eNVM)

2.1.1 Phase Change Memory (PCM)

The feasibility of PCM was first demonstrated by *Charles Sie* in his dissertation [23] using chalcogenide glass. By heating the device differently, PCM can switch between amorphous phase (high resistance state, or HRS) and crystalline phase (low resistance state, or LRS). There are two major research directions of PCM device engineering: The traditional way focuses on searching for viable material alternatives of $Ge_2Sb_2Te_5$ (GST); A later approach, which is called Interfacial Phase Change Memory (IPCM) [24], leverages $GeTe{-}Sb_2Te_3$ superlattice to achieve non-thermal phase changes by simply using laser pulse to change the coordination state of the Germanium atoms. We can carefully control the heating process of a PCM cell to obtain multiple intermediate phases and hence and realize multi-level cell (MLC) design that can store more than one bit on the cell. **Figure 1** shows the cross section view of a PCM cell. The PCM cell structure typically consists of a thin layer of the chalcogenide sandwiched between two inert metal electrodes. Some researchers argue that all two-terminal eNVM devices, including PCM, should be considered as special types of memristors [25], [26].

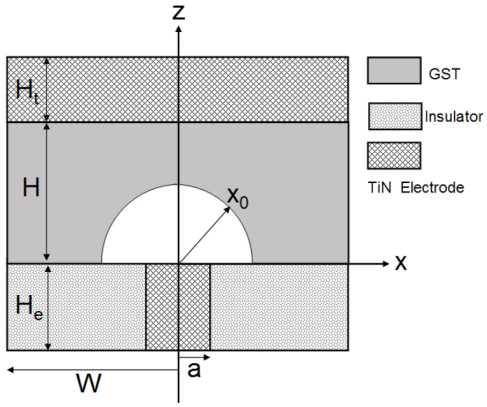


Fig. 1 Cross section view of a PCM cell [27].

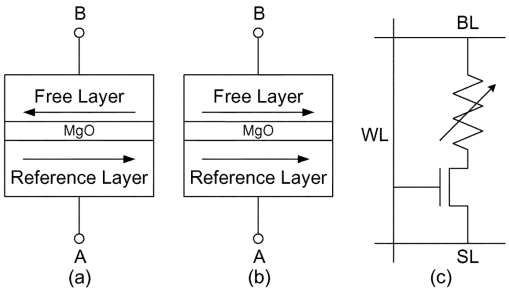


Fig. 2 MTJ structure [28]: (a) Antiparallel (HRS). (b) Parallel (LRS). (c) 1T1J STT-RAM cell structure.

2.1.2 Spin-Transfer Torque Random-Access Memory (STT-RAM)

STT-RAM is a promising eNVM technology that features non-volatility, fast read/write speed (< 10 ns), high programming endurance (> 10^{15} cycles), zero standby power and good scalability [11]. STT-RAM utilizes the spin-transfer torque (STT) effect to switch the resistance of the magnetic tunnel junction (MTJ), which is the storage device of the STT-RAM cell. **Figure 2** shows the basic structure of MTJ and the popular one-transistor-one-MTJ (1T1J) STT-RAM cell structure. In the MTJ, a metal oxide barrier layer (e.g., MgO) is sandwiched between two magnetic layers, namely, reference layer and free layer, respectively. The magnetization of the reference layer is fixed while that of the free layer can be switched by passing a spin-polarized current from different directions. When the magnetization directions of the reference layer and the free layer are in antiparallel (parallel), the MTJ shows as an HRS (LRS). Very recently, binary stochastic synaptic behavior has been demonstrated based on the stochastic switching nature of the MTJ devices at low voltages and programming time-scales [29].

2.1.3 Resistive Random Access Memory (RRAM)

RRAM is one type of eNVMs that is built on dielectric solid-state materials and is often referred to as “memristors”. Based on the formation and annihilation of a conducting region between

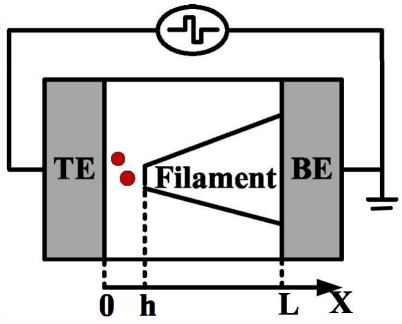


Fig. 3 Metal-oxide memristor [17].

two metallic electrodes separated by an insulating layer, RRAM devices can be generally categorized into two families: 1) oxide-based devices whose resistance change is caused by the displacement of anions into a transition metal oxide, and 2) conductive bridge devices whose resistance change is realized by injecting cations from one active electrode into an electrolyte with a large ionic conductivity and low electronic conductivity. In general, the LRS of a RRAM device is reached through the formation of a conductive pathway between the two electrodes while the HRS (also called "insulating state") is obtained when that pathway is disrupted. The resistance switching process of RRAM devices can be extremely fast (e.g., sub-nanosecond). Many devices also present good retention time (> ten years) and high endurance (> megacycles). **Figure 3** depicts an ion migration filament model of a HfO_x memristor [30]. A HfO_x layer is sandwiched between two metal electrodes TE (top electrode) and BE (bottom electrode). The oxygen ions migrate from the electrode (oxide) interface and recombine with the oxygen vacancies to form a conductive filament between TE and BE.

2.2 Neuromorphic Computing Models

2.2.1 Spiking Neural Network and Spike-timing-dependent Plasticity

In a nervous system, neurons pass electrical and chemical signals to other neurons through synapses. Spike-timing-dependent Plasticity (STDP) is a biologically-observed process for strengthening or weakening these synapse connections [31], which can be considered as an implementation of Hebbian learning [32]. STDP depends on the relative timing between action potentials (spikes) within the input (pre-synaptic) and output (post-synaptic) neurons. In long-term potentiation (LTP), synapses are persistently strengthened by causal spiking, i.e., the pre-synaptic spike occurs before the post-synaptic spike. In long-term depression (LTD), anticausal spiking decreases the synaptic strength. This synaptic plasticity, i.e., the change of synaptic weights, contribute to the essential of how the brain implements learning and inference. Artificial implementation of this spike-based synaptic plasticity is often referred to as Spiking Neural Network (SNN).

One of the most active fields in neuromorphic computing research today is implementing SNNs on CMOS- and eNVM-based circuits. In addition to instantiating neurons and synapses, an SNN often include temporal information in its information encoding and processing schemes [33]. For example, a neuron in an SNN does not fire output signal at end of each propagation cycle as that in classic multilayer perceptron (MLP) networks, but rather fires only when its membrane potential reaches a threshold value. This transmitted signal may also increase or decrease the potentials of the receiving neurons following the rule of STDP. The state of a neuron in an SNN is defined as its current activation level, which is often modeled by differential equations. Various coding methods have been proposed to encode the input using the amplitude and the appearance frequency of the spikes or the temporal relation between the spikes [34], [35]. SNNs are energy-efficient if the spikes are sparsely generated.

2.2.2 Deep Neural Network and Backpropagation

A DNN is an artificial neural network with multiple layers between the input and output layers. A DNN is able to find the underlying relationship between the input and the output of a large volume of data, which usually cannot be explicitly expressed in a mathematical way. Hence, DNNs are particularly capable to model a complex nonlinear relationship. Feedforward network is a typical type of DNNs in which data flows from the input layer to the output layer without loops. Unlike the asynchronous spikes in an SNN, neurons in a DNN output real numbers in a synchronous manner. Neurons in the input layer are fully-connected with the ones in the output layer through synapses which carry synaptic weights. The data moves through the layers of the network to calculate the probability of each output. For example, a DNN that is trained to recognize dog breeds will process the given image and calculate the probability that the dog in the image is a certain breed. The user can review the results and select the probabilities the network should display (e.g., above a certain threshold) and return the classified label(s). During the above process, the network is trained to decompose an image into features at different granularity levels, identify the trends that shared among the training samples, and classify the test samples based on these shared similarities.

Many Variations of DNNs, including convolutional neural networks, deep belief networks, and MLPs, are trained using supervised learning and backpropagation [36], i.e., the backward propagation of errors [37]. In this procedure, errors are computed at the last layer using the loss function and propagate back to the first layer. Weights are updated based on the optimization rule, e.g., gradient descent method iteratively updates the weights along the negative direction of the gradients of the errors [36]. In the recent decade, the application of GPUs greatly shortens the training time of DNNs.

3. CMOS-based NCS

In this section, we introduce the basics of CMOS-based NCS. There exist two distinct philosophies in CMOS-based NCS design: the first one inherits most of the conventional processor design practice and supports state-of-the-art deep learning algorithms; the second one tries to simulate the behaviors of biological neural networks and is often adopted in the study of computational neural science. To show the distinction between these two design philosophies, we first introduce the basic building blocks of synapses and neurons and then introduce the methods to connect these blocks into a large scale system. Finally, we present training and inference schemes of these systems.

3.1 Fundamental Components

3.1.1 Synapse Structure

Von Neumann architecture and near-memory computing. Some DNN accelerators simulate neurons and synapses using von Neumann processors such as conventional CPU/GPUs. The major reason is that the large size of modern DNN models, i.e., more than 10^8 weights of state-of-the-art image recognition models [38], is far beyond the capacity of existing on-chip SRAM. Synaptic weights are stored in an off-chip DRAM and loaded to the processor on demand. Although such a design suffers from low computing efficiency and long memory access latency, it is still the most popular approach to simulate large scale DNNs due to its good scalability and ease usage.

Recent research focus in this area is designing a better memory hierarchy that can minimize the access overhead of the synaptic weights. For example, Chen et al. [39] utilizes a hybrid on-chip memory design composed of embedded DRAM and SRAM to store the synaptic weights and a multiplier array to accelerate the computations, achieving an energy efficiency 150× higher than GPUs. Low-level caches are directly controlled by the compiler to maximize their utilization [40]. These designs are often referred to as "near-memory computing". In some recent designs, the data processing logic is customized according to the computation pattern of the targeted DNN workloads [41].

Processing-in-memory (PIM). Diffrent from von Neumann processors, the "processing-in-memory (PIM)" implementation allows synaptic weights to be stored, computed, and updated inside the memory cells, without any significant data movement. The memory-centric design makes PIM a natural candidate for SNN acceleration. One benefit is that for SNN learning algorithms, the continuous accesses to all synaptic weights can be implemented efficiently in PIM. The hardware architecture of PIM is also close to the structure of a biological neural network. Using 6-transistor SRAM cells to store digital synapses, state-of-the-art implementation achieves a density of 2.1 million single-bit synapses per mm^2 at 14 nm technology node [8]. An alternative approach is to use floating-gate transistors for analog synapses. As a type of a nonvolatile memory, the weights stored in the floating-gate transistors are updated by charging and discharging the gates through current pulses. This process can be utilized to directly simulate the learning rule of SNNs, which makes it especially suitable for SNN acceleration [42], [43].

In CMOS-based DNN accelerator research, the recent focus shift to PIM is mainly triggered by energy efficiency. A SRAM-based DNN accelerator from Biswas et al. [44] achieves up to 100× energy reduction comparing to the discrete memory and computing implementation. However, the scalability of this design is greatly hindered by the low integration density of 6-transistor SRAM cells [45]. The typical on-chip memory size is a few MB, which only allows the execution of MNIST [46]-level applications even binary neural network is applied.

3.1.2 Neurons

In computational neural science study, a variety of neuron circuits were designed to simulate biological neuronal models. Those neuronal models can be as simple as integrate-and-fire which contains only accumulation and threshold detection; or can be as complex as Hodgkin-Huxley model [47] which contains four ordinary differential equations to capture the activation/inactivation of sodium and potassium channels. SNN accelerators often make a good tradesoff between expressiveness and computational complexity of neuronal models. TrueNorth [7], for example, uses reconfigurable circuits to support three types of simplified neuronal models and well approximates Hodgkin-Huxley model by combining these three models. Loihi [8] limits its inference module to leaky integrate-and-fire model, but adds the complex dendrite and axon models to support more flexible learning rules.

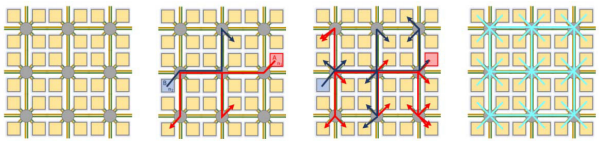


Fig. 4 The mesh network for short-range communication [8].

Implementation of these neuron models can be either mixed-signal or fully digital. For example, Neurogrid [6] directly uses wire capacitance to simulate the integration. Digital design is more commonly used in recent SNN accelerator designs due to its better flexibility and higher performance. Both TrueNorth and Loihi have fully digital synapses and neurons.

On the other hand, although DNNs achieve very high accuracy in many pattern recognition and cognitive tasks, the underlying neuron model is relative simple. Following the McCulloch-Pitts neuron model [48], a typical NCS first computes dot-product of an input vector and a weight vector, and then applies a nonlinear activation function to the output. The corresponding hardware implementation has three stages: 1) multiplier array for element-wise multiplication, 2) adder tree for sum-of-product reduction, and 3) lookup table or analog circuit for nonlinear activation. In some works that are based on binary weights (+1/−1), the multipliers can be removed [49].

3.1.3 Network Architecture

As a bio-inspired network, SNNs may require accurate control at single neuron level. Ideally, the accelerator should support arbitrary communication between millions of neurons, introducing great challenge to the architecture design. The main strategy is to use hierarchical networks [50] that allow high local bandwidth and long-range reachability. Each network level handles the communication between thousands of nodes. In short-range communication, Benjamin et al. uses tree routing network for multicasting [6]. Merolla et al. manage the neurons in blocks and propose a locally synchronous and globally asynchronous design [7]. As shown in **Fig. 4**, Loihi uses unicast mesh network and simulates multicast communication by time multiplexing [8]. A neuron will sequentially connect with its targets in several time step (indicated as red arrows), and a synchronization barrier will be triggered thereafter (indicated as the blue mesh). Long-range communication is typically conducted by auxiliary communication modules. Neural spikes are packed into packages and transmitted through a network interface [51].

In contrast to SNNs, DNNs are based on more coarse-grained networks. A DNN usually contains only tens to hundreds com-

putation nodes (or layers), each of which contains thousands to millions of operations. DNN accelerators usually process only one or a few nodes at each time, regardless of the overall network topology. Each node usually has a very regular computation pattern like matrix multiplication and convolution. As such, DNN accelerators use a synchronous on-chip network to perform single layer execution. For example, Google TPU uses systolic array to accelerate matrix multiplications [40] and Eyeriss uses a specialized row-stationary architecture to accelerate convolutions [41]. Multi-chip parallelization is based on data parallelism, i.e., all accelerators execute the same model on different data. The major component of the communication between accelerators is model synchronization and is independent to DNN's network topology. Zou et al. [52] proposed a tree network for accelerator interconnection.

3.2 Training and Inference Schemes

3.2.1 SNNs

In SNNs, rate coding is often used to convert discrete spikes into continuous values. The average rate of neuronal spikes represents the output value, and the convergence of spiking rate takes multiple cycles of computation. Although the inference operation of SNNs is not as straightforward as that of DNNs, the training of SNNs can be implemented more efficiently using more local learning rules. Spike-timing-dependent Plasticity (STDP) is a classical learning rule for SNNs, i.e.,

$$\Delta w_{ij} = \sum_{k,k'} F\left(t_{ik} - t_{jk'}\right), \tag{1}$$

$$F(\Delta t) = \begin{cases} A_+ \exp\left(-\Delta t/\tau_+\right) & \Delta t > 0 \\ -A_- \exp\left(\Delta t/\tau_-\right) & \Delta t < 0 \end{cases} \tag{2}$$

As shown in Eq. (1), for neuron i and j, the synaptic weight w_{ij} is updated according to the timing difference between pre-synaptic spikes t_{ik} and post-synaptic spikes $t_{jk'}$. Equation (2) gives an example update rule using exponential functions. Notice that each weight update only relies on the information of its adjacent neurons. Such bio-inspired local learning rule introduces much lower hardware overhead of the training circuits than that of the BP algorithm used in DNN training [53].

Although it is hardware friendly, the SNN trained by STDP normally cannot achieve the same level of accuracy of the DNN trained by BP [54]. There have been extensive works searching for better biologically feasible alternatives of BP. The segregated dendrites (SD) model [55] simulates neocortical pyramidal neuron that has two segregated compartments. Park et al. show that a SD-based neuromorphic chip can achieve near-DNN accuracy while keeping a low training overhead, i.e., 7.5% energy consumption overhead comparing to more than 50% overhead in previous works [56].

3.2.2 DNNs

Gradient descent is a popular way to train DNNs. Through BP, all synaptic weights are directly updated to minimized the final loss. For example, we assume a 3-layer DNN $F(x) = f_{w_1}(f_{w_2}(f_{w_3}(x)))$ where x as input and w_i as the layers' weights. Given a loss function $L(F(x))$, the gradients of w_i are computed through the chain rule:

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial f_{w_1}} \frac{\partial f_{w_1}}{\partial w_1}, \tag{3}$$

$$\frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial f_{w_1}} \frac{\partial f_{w_1}}{\partial f_{w_2}} \frac{\partial f_{w_2}}{\partial w_2}, \tag{4}$$

$$\frac{\partial L}{\partial w_3} = \frac{\partial L}{\partial f_{w_1}} \frac{\partial f_{w_1}}{\partial f_{w_2}} \frac{\partial f_{w_2}}{\partial f_{w_3}} \frac{\partial f_{w_3}}{\partial w_3}. \tag{5}$$

Although being effective as an optimization algorithm, BP is not biological feasible and may introduces large hardware cost to support the training process. The main problem of BP is that its computation is not local: calculating the gradient of one network layer requires the activation and weight information of other layers. Therefore, to support training, the hardware must provide some form of information passage that connects distant layers. In the accelerators based on von Neumann architecture, this passage is implemented using large off-chip memory. Intermediate results (i.e., $\partial L/\partial f_{w_1}$ and $\partial f_{w_1}/\partial f_{w_2}$) are stored off-chip and accessed on demand. The direct result of such a design is that GPUs can require over 10 GB of memory to achieve enough training parallelism [57]. Preforming BP on PIM architecture is almost impossible due to the controversy between the limited capacity of SRAM array and the high precision required by the BP process.

4. eNVM-based NCS

In this section, we will discuss the design of eNVM-based NCS, including the fundamental components such as neurons and synapses, the training and inference schemes, and a detailed comparison with CMOS-based NCS. We will show that eNVM-based NCS demonstrate some great potentials in various aspects. We also discuss the challenges of eNVM-based NCS.

4.1 Fundamental Components

4.1.1 Synapses

In an eNVM-based NCS, adjusting a synaptic weight is usually realized by programming the eNVM device between LRS (*SET*) and HRS (*RESET*).

PCM as a synapse. Synaptic weight can be represented by the resistance of a PCM cell, which is programmed by heating the device in different ways. The *SET* process can be implemented incrementally by applying repetitive pulses to slowly crystallize the PCM device. The *RESET* process, however, involves melt and quench and tends to be an abrupt process.

A two-PCM approach was proposed to implement STDP, i.e., using separate devices for LTP and LTD, respectively [58]. When a pre-synaptic neuron spikes, it sends out a read pulse and enters "LTP mode". If the post-synaptic neuron spikes during this period, the LTP synapse receives a partial *SET* pulse. Otherwise, the LTD synapse is programmed. Symmetric and asymmetric STDP were also implemented with a single PCM cell as a synapse [59], where staircase down pulses of varying amplitudes were used for partial *SET*.

STT-RAM as a synapse. STT-RAM uses the current to *RESET* (or *SET*) the polarization of the free ferromagnetic layer. During the *RESET* (*SET*) process, a positive (negative) voltage difference is applied between the source line (SL) and the bit line (BL) in Fig. 2. The current amplitude required to reverse the di-

rection of the free ferromagnetic layer is determined by the size and aspect ratio of MTJ and the write pulse duration.

Vincent et al. used STT-RAM to implement a stochastic memristive synapse by carefully choosing the current and duration time of programming pulses to implement controlled switching probabilities [29]. At device level, binary MTJ is the most common choice due to the mature fabrication process to control the magnetic anisotropy [60]. Multi-bit storage cells can also be realized by either producing more stable states [61] or the stack of MTJs [62].

RRAM as a synapse. During the *RESET* process of a RRAM cell, a partially ruptured conductive filament region with a high resistance per unit length is formed aside the conductive filament region with a low resistance per unit length. The memristor then switches from LRS to HRS. During the *SET* process, the ruptured conductive filament region shrinks, thus switching the memristor from HRS to LRS. Note that the memristor resistance can be programmed to any arbitrary value by applying a programming current with different pulse widths or magnitudes. The memristor resistance changes only when the applied voltage is above a threshold.

Choi et al. reported a gradual *RESET* switching with increasing voltages in a G_dO_x-based RRAM with abrupt *SET* switching [63]. Another approach is to combine the abrupt *SET* operation and binary synaptic devices to implement a stochastic learning rule [64]. Piccolboni et al. reported a HfO_2-based vertical RRAM (VRRAM) technology [65], where each synapse is composed of a stack of RRAMs with one common transistor.

4.1.2 Neurons

Emulation of neuronal dynamics, including the equilibrium potential, the transient dynamics, and the process of neurotransmission, is the key to realizing biologically plausible NCS [66]. However, the complex neuronal dynamics must often be simplified for efficient hardware realizations [67]. The integration of the post-synaptic potentials and the subsequent firing event are the two most important dynamical components. **Figure 5** (a) depicts the schematic of an integrate-and-fire (IF) design featuring high speed and low power consumption [68]. During the operation, the B_L voltage V_y continues increasing until it reaches V_{th}. Then the differential pair ($M_1 - M_4$) together with the following two cascaded inverters ($M_5 - M_7$ and $M_{10} - M_{12}$) generates a high voltage at V_s, which in turn enables the discharging transistor M_{13}. Consequently, V_y decreases quickly and eventually turns off M_{13}. As such, the firing of one output spike at V_{out} is completed and a new iteration of IF starts. Liu et al. implemented and simulated the IF design with IBM 130 nm technology [68]. The waveforms of V_y, V_s, and V_{out} under the fastest firing frequency are shown in Fig. 5 (b).

4.1.3 Crossbar Structure

Dense eNVM crossbar arrays can potentially enable massively parallel and highly energy-efficient neuromorphic computations. An $N \times N$ memristor-based crossbar (MBC) array is illustrated in **Fig. 6** to demonstrate its matrix computation functionality [69]. A set of input voltages $\boldsymbol{V}_I^T = \{v_{I,1}, v_{I,2}, \cdots, v_{I,N}\}$ are applied to the word-lines (WLs) of the array, and collect the current through each bit-line (BL) by measuring the voltage across a sensing resistor. The same sensing resistors are used on all the BLs with resistance r_s, or conductance $g_s = 1/r_s$. The output voltage vector is $\boldsymbol{V}_O^T = \{v_{O,1}, v_{O,2}, \cdots, v_{O,N}\}$. Assume the memristor in the connection between WL_i and BL_j has a memristance of $m_{i,j}$. The corresponding conductance is $g_{i,j} = 1/m_{i,j}$. Then the relationship between the input and output voltages can be expressed by:

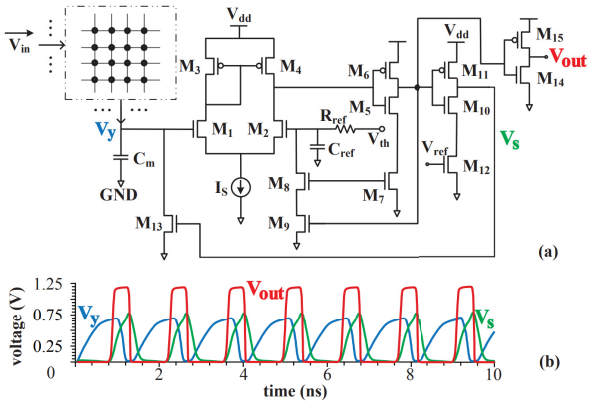


Fig. 5 The integrate-and-fire circuit [68]: (a) the schematic, (b) the simulation waveforms.

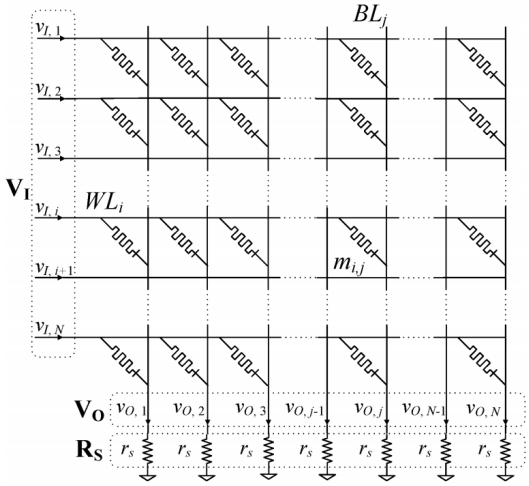


Fig. 6 A memristor crossbar array [69].

$$\boldsymbol{V_O} = \boldsymbol{C} \times \boldsymbol{V_I} \tag{6}$$

Here, $\boldsymbol{C}$ can be represented by the memristances and the load resistors as:

$$\boldsymbol{C} = \boldsymbol{D} \times \boldsymbol{G} = diag(d_1, ..., d_N) \times \begin{bmatrix} g_{1,1} & \cdots & g_{1,N} \\ \vdots & \ddots & \vdots \\ g_{N,1} & \cdots & g_{N,N} \end{bmatrix} \tag{7}$$

where, $d_i = 1/(g_s + \sum_{k=1}^{N} g_{i,k})$.

Sneak path is one of the major concerns in MBC design [72]. This issue refers to as the intrinsic leakage flowing through unselected crossbar cells and degrades the programming efficiency and read correctness of the NCS. The one-transistor-one-resistive device (1T1R) structure, which is shown in **Fig. 7**, can effectively suppress the sneak path leakage and therefore make programming very efficient [73].

The first hardware implementation of an integrated CMOS and memristor crossbar was presented by Lu et al. in Ref. [74]. They

Table 1 A detailed comparison of human brain, GPU, CMOS- and eNVM-based NCS.

System	Type	Neuron	Synapse	# neurons	# synapses	Power	Energy/Connection	Technology	Trainability
Human Brain [9]	Biology	Diverse	Diverse	100 B	10^{15}	20 W	10 fJ	10 μm	Biology
GPU [70]	Digital	-	-	-	-	180 W	7 nJ	16 nm	Any
Neurogrid [6]	Analog	Adaptive Quadratic IF	Shared Dendrite	65 k	100 M	150 mW	100 pJ	180 nm	No
True North [7]	Digital	Adaptive Exponential IF	4-bit Digital	1 M	256 M	72 mW	25 pJ	28 nm	No
Loihi [8]	Digital	Leaky IF	Non-fixed	131 k	126 M	73 mW	23.6 pJ	14 nm	STDP
PCM [71]	Analog	Leaky IF	Analog Weight	256	64 k	-	0.9 pJ	90 nm	STDP

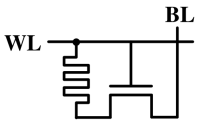


Fig. 7 1T1R design in a crossbar cell [68].

demonstrated a high yielding 40×40 crossbar array controlled by a CMOS platform. A variant of backpropagation algorithm [75] was used to train a three-layer perceptron network with 164,885 synapses on a subset (5,000 images) of MNIST. A 32×32 MBC was used to demonstrate parallel learning for pattern classification problem [68]. This work realized a BSB recall circuit under physical constraints and discussed about the impact of random noises.

4.2 Training and Inference Schemes

4.2.1 STDP for SNNs

The mapping of STDP as a local learning rule in eNVM arrays is very straightforward. It only needs to modify the resistance of eNVM based on the timing of spikes from pre- and post-synaptic neurons. Some studies have been performed in simulation using parameters extracted from biological observations. For example, Diehl et al. demonstrated 95% accuracy on MNIST by utilizing STDP along with other biologically plausible features such as lateral inhibition and adaptive spiking threshold [76].

STDP was implemented with eNVM devices in many prior arts. Chen et al. employed the symmetric/asymmetric memristors and the simplified neurons to perform the STDP learning ability on a 25×25 array [77]. To implement an array design and address the sneak-path issue, Kim et al. proposed a 2T1R synaptic cell for STDP learning [71]. The authors demonstrated a design with 256 neurons and 64 k PCM synaptic cells, on-chip leaky integrate-and-fire neuron, and STDP circuits for on-line and continuous training.

4.2.2 Multiply-accumulate Operation for DNNs

Arrays of analog resistive memory are ideally suited for the multiply-accumulate (MAC) operations which constitute the majority of computations in DNN inference and training. The multiply operation is performed at every cross-point by Ohm's law and the current summation along each column follows Kirchhoff's current law. These MAC operations can be performed in parallel at the stored location of the data with local analog computing and hence, avoid the power consumption of data movement. Different learning rules such as gradient descent can be implemented on resistive crossbar arrays. Alibart et al. presented a small-scale pattern classification task using a 2×9 crossbar array and delta learning rule [78]. Gamrat et al. applied the spike-coding for inference, showing competitive performance on MNIST with pre-trained weights stored on memristive devices [79]. Training and inference of a small one-layer DNN were implemented on a 12×12 memristor crossbar requiring no separate selection device [80]. Hassan et al. proposed a hybrid spiking-based multi-layered NCS that can perform online training [81].

4.3 Comparing CMOS- and eNVM-based NCS

Table 1 shows a comparison of a typical human brain, an Nvidia Geforce GTX 1080 GPU, CMOS- and eNVM-based NCS, on speed, power, and other critical aspects. Some of the key components in the comparison are summarized and discussed below:

Neuron represents the model of neurons implemented in the system. Typically, a variation of the IF design is adopted.

Synapse can be constructed by digital devices or eNVMs. The numbers of neurons and synapses are also listed in the table. Due to the limitation of device reliability, eNVM-based systems usually possess fewer synapses than CMOS-based systems. And all artificial systems have far fewer synapses than a human brain (i.e., 10^6 vs. 10^{15}).

The energy per connection. The energy per connection is the average energy required to pass one spike through one synapse or to compute one floating point operation. Since synaptic processing dominates the system's energy cost, this portion of the energy contributes to the majority of the energy consumption of the system. eNVM-based NCS, on the other hand, achieves less than 1 pJ on energy/connection, which is considerably lower than that of CMOS-based ones.

4.4 Challenges

Although eNVM-based NCS demonstrated great potentials in neuromorphic computing, there still exist some challenges in the design, including:

1) **Reliability**. Unlike the mature Silicon CMOS technology, one of the major drawbacks of most emerging memory technologies is their limited reliability characteristics. Many of these devices suffer from large device-to-device variability in their operations and also exhibit cycle-to-cycle variability in the programming of a single device. The programmed resistance levels also suffer from drift and retention loss, which become more phenomenal when technology scales. For example, the randomness of filament formation in the memrisor [82] can be magnified by the nonlinear dynamics of memrisor switching [83]. Circuit and architectural solutions become essential to overcome the reliability issue of the eNVM-based NCS on top of the traditional device

engineering solutions.

2) **Integration**. The integration of MBC and CMOS-based neuron circuits also generate many unique design challenges. For example, the interconnect IR-drop in memristor arrays severely limits the scale of MBCs and hence hinders the design scalability. Liu et al. showed that both reading (recall) and writing (training) of the MBC will encounter severe reliability issues when the array size is beyond 64×64 [18]. In addition, the fabrication process of memristor devices is still under development, thus still achieving unsatisfying yield at the array level. Stuck-on and stuck-off defects are often observed [84]. Also, due to the intrinsic mechanism of the memristor, analog memristance values are vulnerable to read disturbance, that is, the memristance could drift from its originally programmed value after a number of accesses. Therefore, an real-time feedback controller might be needed to trace and control such drift.

3) **Security**. Neuromorphic computing are widely used in many applications for advanced data processing, and often implements proprietary algorithms. While DNNs have achieved remarkable success in advanced intelligent applications, security-related issues gradually become severe. For example, the learning model running on an embedded device is exposed to the risk of being attacked by malicious users who have physical access to the device [20]. Attackers can also send hand-crafted data into the system and achieve anticipated purposes, e.g., to fool the model's predictions in testing phase using adversarial attack [85] or to compromise the model in training phase using poisoning attack [86].

5. EDA-based Solutions

In this section, we will discuss some EDA-based solutions to address the design challenges of eNVM-based NCS, including reliability, precision, scalability, and security.

5.1 Reliability and Integration

Programming a memristor to a specific state can be very challenging because real-time monitoring the memristor state is not practical [87]. Liu et al. proposed a noise-eliminating training method over a new crossbar structure to minimize the noise accumulation during the MBC training and improve the trained system performance, i.e., the pattern recall rate [17]. The sensitivity of the MBC programming to the process variations and input signal noise was quantitatively analyzed. A digital-assisted initialization step for MBC training was also introduced to reduce the training failure rate as well as the training time. Experiment results were evaluated on multiple 512×512 MBC computing engines. Xia et al. proposed a fault-tolerant online training method that alternates between two phases [88]. The fault-detection phase used a quiescent-voltage comparison method to obtain faulty cells during training. The fault-tolerant phase used a threshold-training method and a re-mapping scheme to tolerate the faults.

Another issue of MBCs is that the voltage applied to the two terminals of a memristor is affected by the device location in the crossbar and the resistance states of all other memristors. Liu et al. proposed a novel system reduction scheme that significantly lowers the required dimension of the memristor crossbars in NCS while maintaining high computing accuracy [18]. An IR-drop compensation technique was also proposed to overcome the adverse impacts of the wire resistance and the sneak-path problem in large memristor crossbar designs [18]. Another approach is to partition a large network into many small ones in lower dimension [89].

The training of NCS can be further improved if the realistic hardware limits are taken into consideration. Liu et al. presented a quantitative analysis on the impact of device imperfections and circuit design constraints and proposed a novel variation-aware training scheme to enhance the training robustness of MBC-based NCS by actively compensating the impact of device variations and optimizing the mapping scheme from computations to crossbars [19]. Most eNVM devices show nonlinear and asymmetric switching behaviors. These characteristics may prevent backpropogation algorithm from locating optimal weights during training [90]. Such nonlinear behaviors also degrade the system tolerance to noises. Gong et al. established a practical methodology based on Gaussian process regression to address this issue [91]. The proposed methodology is independent on switching mechanisms and applicable to various eNVM devices.

5.2 Precision

The performance of eNVM-based NCS usually depends on how accurately the state of the devices can be programmed. Alibart et al. designed a simple write algorithm to tune device conductance at a specific bias point within its dynamic range even in the presence of large variations in switching behavior [92]. Except for device engineering optimization, algorithm-level technique is also helpful to relax the requirement of the resistance state resolution. By modifying the regularization in training, the weight distribution can be tuned to fit the resistance values [93]. In this way, algorithm-level weights are tailored to enhance the accuracy of eNVM-based NCS.

To better alleviate the impact of device variations, binarized neural networks (BNNs) can be implemented in eNVM-based NCS. BNNs only allow binary synapses of $+1$ and -1. Compare to floating point DNNs, BNNs avoids the complex matrix multiplications by leveraging simple binary operations [94]. Pham et al. proposed a Memristor-CMOS hybrid circuits of BNNs, where single-column and double-column memristor BNNs were presented and activation function are realized using CMOS circuits [95]. Experiment results showed that the memristor BNNs could recognize as many as 90% MNIST digits when the memristance variation is as large as 25%. Additionally, Nandakumar et al. proposed a mixed-precision architecture that combines a computational memory unit storing the synaptic weights with a digital processing unit and an additional memory unit that stored the accumulated weight updates in high precision [96].

5.3 Scalability

To further increase the design density of NCS, various works are proposed to implement 3D structured eNVMs. Researchers from Intel demonstrated the ability to stack multiple layers of PCM arrays within a single die [97]. To alleviate the write overhead of STT-RAM in a 3D multi-core environment, Mishra et

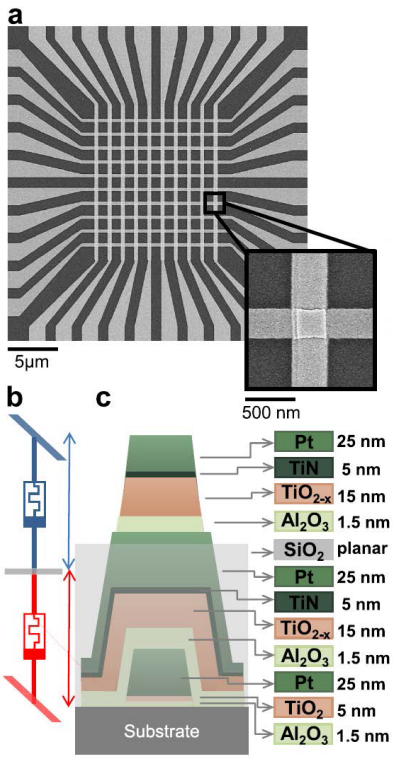


Fig. 8 Fabrication details [101]. (a) Scanning electron microscopy top-view image of the fabricated circuit with a zoom on a stacked device to highlight the clean electrode edges. (b) Equivalent circuit for two memristors in the stacked configuration, in particular, highlighting that the middle electrode (gray) is shared between bottom (red) and top (blue) devices. (c) Cartoon of device's cross section showing the material layers and their corresponding thicknesses.

al. proposed a network-level solution to prioritize cache accesses to the idle STT-RAM cache banks [98]. Zhang et al. systematically analyzed the variation sources of multi-level cell (MLC) STT-RAM designs and their impacts on the reliability of the read and write operations [62]. There also have already been demonstrations of multilayer crossbar circuits, whose primary target application was digital memories [99], [100]. In particular, Adam et al. [101] reported a monolithically integrated 3-D metal-oxide memristor crossbar circuit suitable for analog neuromorphic computing applications. The demonstrated crossbar was based on $Pt/Al_2O_3/TiO_{2-x}/TiN/Pt$ memristors and consisted of a stack of two 10×10 crossbars with shared middle electrodes. The results demonstrated a significant improvement of yield and uniformity of crossbar devices. This fabrication process could also be integrated with CMOS circuits. **Figure 8** demonstrates two memristive crossbar circuits are monolithically integrated in a conventional configuration, with middle lines shared between the bottom and top crossbar circuits.

5.4 Security

Running DNN models on an embedded device has a unique security challenge that the learning model can be replicated by attackers who have access to the system by learning from the input/output pairs [20]. Yang et al. [20] proposed a secure memristor-based NCS design to thwart such replication attacks by leveraging memristor's obsolescence effect. The accuracy of the system outputs for any unauthorized user was reduced. For a legitimate user, the obsolescence effect was regulated, thereby the accuracy of the outputs can be maintained at a high level. The proposed methodology was compatible with mainstream classification applications, memristor devices, and security and performance constraints.

Very recently, adversarial attack [85] become a significant concern. State-of-the-art defenses against adversarial attack involve adversarial example detection via multi-model cross verification, followed by adversarial example filtration. Although this procedure has been proved effective, the high computational overhead and considerable input data loss make this solution hard to deploy in reality. To overcome the above drawbacks, Liu et al. [21] proposed a novel adversarial example restoration system to restore the adversarially perturbed input to its original state. It included a restoration network based on residual learning and a hardware implementation by leveraging neuromorphic technique to achieve an effective and efficient defense. The proposed restoration system demonstrates a high restoration rate that outperforms state-of-the-art methods with high image quality. The restoration system can be easily integrated into the existing NCS.

6. Future of NCS

As aforementioned, NCS can greatly benefit from eNVM technologies from many design aspects. On the one hand, the high computing efficiency of modern NCS designs and the advance in computing architecture motivate explorations on new algorithms that better fit the NCS structures; on the other hand, new algorithms will also promote the study of novel NCS architectures that can run these algorithms more efficiently. NCS will also be expected to offer much higher performance than conventional von Neumann systems for specific tasks such as image/pattern recognition/prediction and natural language processing. The recent TrueNorth chip from IBM is a perfect illustration of such a strategy where various machine learning components were integrated exclusively using low-power CMOS technology. The energy efficiency and integration density of such chips could be enhanced significantly by combining emerging memory based cross-bar arrays with CMOS circuitry that mimics neuronal activities. More importantly, these new devices could play a definitive role in enabling online learning in large cognitive computing systems. In addition, many of these new implementations try to benefit from the intrinsic stochasticity and variability of emerging memories.

7. Conclusion

In this paper, we introduced the CMOS- and eNVM-based NCS from many aspects. We first presented the implementations of the fundamental components, including both neurons and synapses. We then compared training and inference schemes of different neuromorphic computing algorithms such as DNNs and SNNs. The limitations of CMOS-based designs in scalability and connectivity were discussed and the potentials of eNVM-based designs were analyzed in details. We also discuss various design solutions for the design challenges of eNVM-based NCS in reliability, integration, and security. Finally, we provided our insights on the future of NCS and predicted the trend of integrating CMOS circuitry and emerging devices.

Acknowledgments This work was supported by NSF-1725456, NSF-1822085, and the memberships of NSF IUCRC for ASIC from Horizon Robotics, Unisound, etc.

References

[1] Backus, J.: *Can programming be liberated from the von Neumann style? A functional style and its algebra of programs*, ACM (2007).

[2] Mead, C.: Neuromorphic electronic systems, *Proc. IEEE*, Vol.78, No.10, pp.1629–1636 (1990).

[3] Sterling, P. and Laughlin, S.: *Principles of neural design*, MIT Press (2015).

[4] Farquhar, E., Gordon, C. and Hasler, P.: A field programmable neural array, *IEEE International Symposium on Circuits and Systems* (*ISCAS*), pp.4114–4117 (2006).

[5] Poon, C.-S. and Zhou, K.: Neuromorphic silicon neurons and large-scale neural networks: Challenges and opportunities, *Frontiers in Neuroscience*, Vol.5, p.108 (2011).

[6] Benjamin, B.V., Gao, P., McQuinn, E., Choudhary, S., Chandrasekaran, A.R., Bussat, J.-M., Alvarez-Icaza, R., Arthur, J.V., Merolla, P.A. and Boahen, K.: Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations, *Proc. IEEE*, Vol.102, No.5, pp.699–716 (2014).

[7] Merolla, P.A., Arthur, J.V., Alvarez-Icaza, R., Cassidy, A.S., Sawada, J., Akopyan, F., Jackson, B.L., Imam, N., Guo, C., Nakamura, Y., et al.: A million spiking-neuron integrated circuit with a scalable communication network and interface, *Science*, Vol.345, No.6197, pp.668–673 (2014).

[8] Davies, M., Srinivasa, N., Lin, T.-H., Chinya, G., Cao, Y., Choday, S.H., Dimou, G., Joshi, P., Imam, N., Jain, S., et al.: Loihi: A neuromorphic manycore processor with on-chip learning, *IEEE Micro*, Vol.38, No.1, pp.82–99 (2018).

[9] Furber, S.: Large-scale neuromorphic computing systems, *Journal of Neural Engineering*, Vol.13, No.5, p.051001 (2016).

[10] Wong, H.-S.P., Raoux, S., Kim, S., Liang, J., Reifenberg, J.P., Rajendran, B., Asheghi, M. and Goodson, K.E.: Phase change memory, *Proc. IEEE*, Vol.98, No.12, pp.2201–2227 (2010).

[11] Hosomi, M., Yamagishi, H., Yamamoto, T., Bessho, K., Higo, Y., Yamane, K., Yamada, H., Shoji, M., Hachino, H., Fukumoto, C., et al.: A novel nonvolatile memory with spin torque transfer magnetization switching: Spin-RAM, *IEEE International Electron Devices Meeting* (*IEDM*), pp.459–462 (2005).

[12] 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.: Metal–oxide RRAM, *Proc. IEEE*, Vol.100, No.6, pp.1951–1970 (2012).

[13] Burr, G.W., Shelby, R.M., Sebastian, A., Kim, S., Kim, S., Sidler, S., Virwani, K., Ishii, M., Narayanan, P., Fumarola, A., et al.: Neuromorphic computing using non-volatile memory, *Advances in Physics: X*, Vol.2, No.1, pp.89–124 (2017).

[14] Liu, X., Mao, M., Liu, B., Li, H., Chen, Y., Li, B., Wang, Y., Jiang, H., Barnell, M., Wu, Q., et al.: RENO: A high-efficient reconfigurable neuromorphic computing accelerator design, *ACM/EDAC/IEEE Design Automation Conference* (*DAC*), pp.1–6 (2015).

[15] Song, L., Qian, X., Li, H. and Chen, Y.: Pipelayer: A pipelined reram-based accelerator for deep learning, *IEEE International Symposium on High Performance Computer Architecture* (*HPCA*), pp.541–552 (2017).

[16] Chen, F., Song, L. and Chen, Y.: ReGAN: A pipelined ReRAM-based accelerator for generative adversarial networks, *Asia and South Pacific Design Automation Conference* (*ASP-DAC*), pp.178–183 (2018).

[17] Liu, B., Hu, M., Li, H., Mao, Z.-H., Chen, Y., Huang, T. and Zhang, W.: Digital-assisted noise-eliminating training for memristor crossbar-based analog neuromorphic computing engine, *ACM/EDAC/IEEE Design Automation Conference* (*DAC*), pp.1–6 (2013).

[18] Liu, B., Li, H., Chen, Y., Li, X., Huang, T., Wu, Q. and Barnell, M.: Reduction and IR-drop compensations techniques for reliable neuromorphic computing systems, *IEEE/ACM International Conference on Computer-Aided Design* (*ICCAD*), pp.63–70 (2014).

[19] Liu, B., Li, H., Chen, Y., Li, X., Wu, Q. and Huang, T.: Vortex: Variation-aware training for memristor x-bar, *ACM/EDAC/IEEE Design Automation Conference* (*DAC*), pp.1–6 (2015).

[20] Yang, C., Liu, B., Li, H., Chen, Y., Barnell, M., Wu, Q., Wen, W. and Rajendran, J.: Security of neuromorphic computing: Thwarting learning attacks using memristor's obsolescence effect, *IEEE/ACM International Conference on Computer-Aided Design* (*ICCAD*), pp.1–6 (2016).

[21] Liu, Y., Xie, Y. and Srivastava, A.: Neural trojans, *IEEE International Conference on Computer Design* (*ICCD*), pp.45–48 (2017).

[22] Liu, B., Yang, C., Li, H., Chen, Y., Wu, Q. and Barnell, M.: Security of neuromorphic systems: Challenges and solutions, *IEEE International Symposium on Circuits and Systems* (*ISCAS*), pp.1326–1329 (2016).

[23] Sie, C.: Memory devices using bistable resistivity in amorphous As-Te-Ge films, PhD Thesis, Ph.D. dissertation, Iowa State University, Ames, IA (1969).

[24] Simpson, R., Fons, P., Kolobov, A., Fukaya, T., Krbal, M., Yagi, T. and Tominaga, J.: Interfacial phase-change memory, *Nature Nanotechnology*, Vol.6, No.8, p.501 (2011).

[25] Chua, L.: Resistance switching memories are memristors, *Applied Physics A*, Vol.102, No.4, pp.765–783 (2011).

[26] Mellor, C.: HP and Hynix to produce the memristor goods by 2013, The Register (2012).

[27] Rajendran, B., Karidis, J., Lee, M., Breitwisch, M., Burr, G., Shih, Y., Cheek, R., Schrott, A., Lung, H. and Lam, C.: Analytical model for reset operation of phase change memory, *IEEE International Electron Devices Meeting* (*IEDM*), pp.1–4 (2008).

[28] Zhang, Y., Wang, X., Li, H. and Chen, Y.: STT-RAM cell optimization considering MTJ and CMOS variations, *IEEE Trans. Magnetics*, Vol.47, No.10, pp.2962–2965 (2011).

[29] Vincent, A.F., Larroque, J., Locatelli, N., Romdhane, N.B., Bichler, O., Gamrat, C., Zhao, W.S., Klein, J.-O., Galdin-Retailleau, S. and Querlioz, D.: Spin-transfer torque magnetic memory as a stochastic memristive synapse for neuromorphic systems, *IEEE Trans. Biomedical Circuits and Systems* (*TBioCAS*), Vol.9, No.2, pp.166–174 (2015).

[30] Yu, S., Wu, Y. and Wong, H.-S.P.: Investigating the switching dynamics and multilevel capability of bipolar metal oxide resistive switching memory, *Applied Physics Letters*, Vol.98, No.10, p.103514 (2011).

[31] Taylor, M.: The problem of stimulus structure in the behavioural theory of perception, *S. Afr. J. Psychol*, Vol.3, pp.23–45 (1973).

[32] Hebb, D.O.: *The organization of behavior; a neuropsycholocigal theory*, John Wiley & Sons Inc. (1949).

[33] Maass, W.: Networks of spiking neurons: The third generation of neural network models, *Neural Networks*, Vol.10, No.9, pp.1659–1671 (1997).

[34] Zambrano, D., Nusselder, R., Scholte, H.S. and Bohte, S.: Efficient computation in adaptive artificial spiking neural networks, arXiv preprint arXiv:1710.04838 (2017).

[35] Rueckauer, B. and Liu, S.-C.: Conversion of analog to spiking neural networks using sparse temporal coding, *IEEE International Symposium on Circuits and Systems* (*ISCAS*), pp.1–5 (2018).

[36] Goodfellow, I., Bengio, Y. and Courville, A.: *Deep learning*, MIT press (2016).

[37] Rumelhart, D.E., Hinton, G.E. and Williams, R.J.: Learning representations by back-propagating errors, *Nature*, Vol.323, No.9 (1986).

[38] Cheng, Y., Wang, D., Zhou, P. and Zhang, T.: A survey of model compression and acceleration for deep neural networks, arXiv preprint arXiv:1710.09282 (2017).

[39] Chen, Y., Luo, T., Liu, S., Zhang, S., He, L., Wang, J., Li, L., Chen, T., Xu, Z., Sun, N., et al.: Dadiannao: A machine-learning supercomputer, *IEEE/ACM International Symposium on Microarchitecture* (*MICRO*), pp.609–622 (2014).

[40] Jouppi, N.P., Young, C., Patil, N., Patterson, D., Agrawal, G., Bajwa, R., Bates, S., Bhatia, S., Boden, N., Borchers, A., et al.: In-datacenter performance analysis of a tensor processing unit, *IEEE/ACM International Symposium on Computer Architecture* (*ISCA*), pp.1–12 (2017).

[41] Chen, Y.-H., Krishna, T., Emer, J.S. and Sze, V.: Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks, *IEEE Journal of Solid-State Circuits* (*JSSC*), Vol.52, No.1, pp.127–138 (2017).

[42] Yang, H., Sheu, B.J. and Lee, J.-C.: A nonvolatile analog neural memory using floating-gate MOS transistors, *Analog Integrated Circuits and Signal Processing*, Vol.2, No.1, pp.19–25 (1992).

[43] Hindo, T.: Weight updating floating-gate synapse, *Electronics Letters*, Vol.50, No.17, pp.1190–1191 (2014).

[44] Biswas, A. and Chandrakasan, A.P.: Conv-RAM: An energy-efficient SRAM with embedded convolution computation for low-power CNN-based machine learning applications, *IEEE International Solid-State Circuits Conference* (*ISSCC*), pp.488–490 (2018).

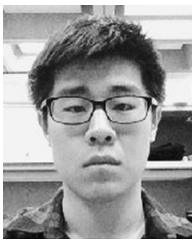
[45] Zhang, J., Wang, Z. and Verma, N.: A machine-learning classifier implemented in a standard 6T SRAM array, *IEEE Symposium on VLSI Circuits* (*VLSI-Circuits*), pp.1–2 (2016).

[46] LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., et al.: Gradient-based learning applied to document recognition, *Proc. IEEE*, Vol.86, No.11, pp.2278–2324 (1998).

[47] Hodgkin, A.L. and Huxley, A.F.: A quantitative description of membrane current and its application to conduction and excitation in

nerve, *The Journal of Physiology*, Vol.117, No.4, pp.500–544 (1952).
[48] McCulloch, W.S. and Pitts, W.: A logical calculus of the ideas immanent in nervous activity, *Bulletin of Mathematical Biology*, Vol.52, No.1-2, pp.99–115 (1990).
[49] Andri, R., Cavigelli, L., Rossi, D. and Benini, L.: YodaNN: An architecture for ultralow power binary-weight CNN acceleration, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (*TCAD*), Vol.37, No.1, pp.48–60 (2018).
[50] Furber, S.B., Lester, D.R., Plana, L.A., Garside, J.D., Painkras, E., Temple, S. and Brown, A.D.: Overview of the spinnaker system architecture, *IEEE Trans. Computers*, Vol.62, No.12, pp.2454–2467 (2013).
[51] Wu, J., Furber, S. and Garside, J.: A programmable adaptive router for a GALS parallel system, *2009 15th IEEE Symposium on Asynchronous Circuits and Systems*, pp.23–31 (2009).
[52] Zou, Y., Jin, X., Li, Y., Guo, Z., Wang, E. and Xiao, B.: Mariana: Tencent deep learning platform and its applications, *Proc. VLDB Endowment*, Vol.7, No.13, pp.1772–1777 (2014).
[53] Schemmel, J., Briiderle, D., Griibl, A., Hock, M., Meier, K. and Millner, S.: A wafer-scale neuromorphic hardware system for large-scale neural modeling, *IEEE International Symposium on Circuits and Systems* (*ISCAS*), pp.1947–1950 (2010).
[54] Kheradpisheh, S.R., Ganjtabesh, M., Thorpe, S.J. and Masquelier, T.: STDP-based spiking deep convolutional neural networks for object recognition, *Neural Networks*, Vol.99, pp.56–67 (2018).
[55] Guerguiev, J., Lillicrap, T.P. and Richards, B.A.: Towards deep learning with segregated dendrites, *eLife*, Vol.6, p.e22901 (2017).
[56] Park, J., Lee, J. and Jeon, D.: 7.6 A 65nm 236.5 nJ/Classification Neuromorphic Processor with 7.5% Energy Overhead On-Chip Learning Using Direct Spike-Only Feedback, *IEEE International Solid-State Circuits Conference* (*ISSCC*), pp.140–142 (2019).
[57] Meng, C., Sun, M., Yang, J., Qiu, M. and Gu, Y.: Training deeper models by GPU memory optimization on TensorFlow, *ML Systems Workshop in NIPS* (2017).
[58] Suri, M., Bichler, O., Querlioz, D., Cueto, O., Perniola, L., Sousa, V., Vuillaume, D., Gamrat, C. and DeSalvo, B.: Phase change memory as synapse for ultra-dense neuromorphic systems: Application to complex visual pattern extraction, *International Electron Devices Meeting* (*IEDM*), pp.4.4.1–4.4.4 (2011).
[59] Kuzum, D., Jeyasingh, R.G. and Wong, H.-S.P.: Energy efficient programming of nanoelectronic synaptic devices for large-scale implementation of associative and temporal sequence learning, *International Electron Devices Meeting* (*IEDM*), pp.30.3.1–30.3.4 (2011).
[60] Yan, B., Chen, F., Zhang, Y., Song, C., Li, H. and Chen, Y.: Exploring the opportunity of implementing neuromorphic computing systems with spintronic devices, *Design, Automation & Test in Europe Conference & Exhibition* (*DATE*), pp.109–112 (2018).
[61] Stainer, Q., Lombard, L., Mackay, K., Lee, D., Bandiera, S., Portemont, C., Creuzet, C., Sousa, R. and Dieny, B.: Self-referenced multi-bit thermally assisted magnetic random access memories, *Applied Physics Letters*, Vol.105, No.3, p.032405 (2014).
[62] Zhang, Y., Zhang, L., Wen, W., Sun, G. and Chen, Y.: Multi-level cell STT-RAM: Is it realistic or just a dream?, *IEEE/ACM International Conference on Computer-Aided Design* (*ICCAD*), pp.526–532 (2012).
[63] Choi, H., Jung, H., Lee, J., Yoon, J., Park, J., Seong, D.-J., Lee, W., Hasan, M., Jung, G.-Y. and Hwang, H.: An electrically modifiable synapse array of resistive switching memory, *Nanotechnology*, Vol.20, No.34, p.345201 (2009).
[64] Yu, S., Gao, B., Fang, Z., Yu, H., Kang, J. and Wong, H.-S.P.: Stochastic learning in oxide binary synaptic device for neuromorphic computing, *Frontiers in Neuroscience*, Vol.7, p.186 (2013).
[65] Piccolboni, G., Molas, G., Portal, J., Coquand, R., Bocquet, M., Garbin, D., Vianello, E., Carabasse, C., Delaye, V., Pellissier, C., et al.: Investigation of the potentialities of Vertical Resistive RAM (VRRAM) for neuromorphic applications, *IEEE International Electron Devices Meeting* (*IEDM*), pp.17.2.1–17.2.4 (2015).
[66] Gerstner, W., Kistler, W.M., Naud, R. and Paninski, L.: *Neuronal dynamics: From single neurons to networks and models of cognition*, Cambridge University Press (2014).
[67] Indiveri, G., Linares-Barranco, B., Hamilton, T.J., Van Schaik, A., Etienne-Cummings, R., Delbruck, T., Liu, S.-C., Dudek, P., Häfliger, P., Renaud, S., et al.: Neuromorphic silicon neuron circuits, *Frontiers in Neuroscience*, Vol.5, p.73 (2011).
[68] Liu, C., Yan, B., Yang, C., Song, L., Li, Z., Liu, B., Chen, Y., Li, H., Wu, Q. and Jiang, H.: A spiking neuromorphic design with resistive crossbar, *ACM/EDAC/IEEE Design Automation Conference* (*DAC*), pp.1–6 (2015).
[69] Hu, M., Li, H., Wu, Q., Rose, G.S. and Chen, Y.: Memristor crossbar based hardware realization of BSB recall function, *International Joint Conference on Neural Networks* (*IJCNN*), pp.1–7 (2012).
[70] *Nvidia Geforce GTX 1080*. available from ⟨https://www.nvidia.com/en-us/geforce/products/10series/geforce-gtx-1080/⟩, (accessed 2019-03-14).
[71] Kim, S., Ishii, M., Lewis, S., Perri, T., BrightSky, M., Kim, W., Jordan, R., Burr, G., Sosa, N., Ray, A., et al.: NVM neuromorphic core with 64k-cell (256-by-256) phase change memory synaptic array with on-chip neuron circuits for continuous in-situ learning, *IEEE International Electron Devices Meeting* (*IEDM*), pp.17.1.1–17.1.4 (2015).
[72] Flocke, A. and Noll, T.G.: Fundamental analysis of resistive nanocrossbars for the use in hybrid Nano/CMOS-memory, *European Solid-State Circuits Conference* (*ESSCIRC*), pp.328–331 (2007).
[73] Huang, J.-J., Tseng, Y.-M., Luo, W.-C., Hsu, C.-W. and Hou, T.-H.: One selector-one resistor (1S1R) crossbar array for high-density flexible memory applications, *International Electron Devices Meeting* (*IEDM*), pp.31.7.1–31.7.4 (2011).
[74] Kim, K.-H., Gaba, S., Wheeler, D., Cruz-Albrecht, J.M., Hussain, T., Srinivasa, N. and Lu, W.: A functional hybrid memristor crossbar-array/CMOS system for data storage and neuromorphic applications, *Nano Letters*, Vol.12, No.1, pp.389–395 (2011).
[75] Burr, G.W., Shelby, R.M., Sidler, S., di Nolfo, C., Jang, J., Boybat, I., Shenoy, R.S., Narayanan, P., Virwani, K., Giacometti, E.U., Kurdi, B.N. and Hwang, H.: Experimental Demonstration and Tolerancing of a Large-Scale Neural Network (165 000 Synapses) Using Phase-Change Memory as the Synaptic Weight Element, *IEEE Trans. Electron Devices* (*T-ED*), Vol.62, No.11, pp.3498–3507 (2015).
[76] Diehl, P.U. and Cook, M.: Unsupervised learning of digit recognition using spike-timing-dependent plasticity, *Frontiers in Computational Neuroscience*, Vol.9, p.99 (2015).
[77] Chen, L., Li, C., Huang, T., He, X., Li, H. and Chen, Y.: STDP learning rule based on memristor with STDP property, *International Joint Conference on Neural Networks* (*IJCNN*), pp.1–6 (2014).
[78] Alibart, F., Zamanidoost, E. and Strukov, D.B.: Pattern classification by memristive crossbar circuits using ex situ and in situ training, *Nature Communications*, Vol.4, p.2072 (2013).
[79] Gamrat, C., Bichler, O. and Roclin, D.: Memristive based device arrays combined with spike based coding can enable efficient implementations of embedded neuromorphic circuits, *IEEE International Electron Devices Meeting* (*IEDM*), pp.4.5.1–4.5.7 (2015).
[80] Burr, G., Narayanan, P., Shelby, R., Sidler, S., Boybat, I., di Nolfo, C. and Leblebici, Y.: Large-scale neural networks implemented with non-volatile memory as the synaptic weight element: Comparative performance analysis (accuracy, speed, and power), *IEEE International Electron Devices Meeting* (*IEDM*), pp.4.4.1–4.4.4 (2015).
[81] Hassan, A.M., Yang, C., Liu, C., Li, H.H. and Chen, Y.: Hybrid spiking-based multi-layered self-learning neuromorphic system based on memristor crossbar arrays, *Design, Automation & Test in Europe Conference & Exhibition* (*DATE*), pp.776–781 (2017).
[82] Liu, Q., Long, S., Lv, H., Wang, W., Niu, J., Huo, Z., Chen, J. and Liu, M.: Controllable growth of nanoscale conductive filaments in solid-electrolyte-based ReRAM by using a metal nanocrystal covered bottom electrode, *ACS Nano*, Vol.4, No.10, pp.6162–6168 (2010).
[83] Chua, L.O.: Local activity is the origin of complexity, *International Journal of Bifurcation and Chaos* (*IJBC*), Vol.15, No.11, pp.3435–3456 (2005).
[84] Liu, C., Hu, M., Strachan, J.P. and Li, H.: Rescuing memristor-based neuromorphic design with high defects, *ACM/EDAC/IEEE Design Automation Conference* (*DAC*), pp.1–6 (2017).
[85] Goodfellow, I.J., Shlens, J. and Szegedy, C.: Explaining and harnessing adversarial examples, *International Conference on Learning Representations* (*ICLR*) (2015).
[86] Yang, C., Wu, Q., Li, H. and Chen, Y.: Generative poisoning attack method against neural networks, arXiv preprint arXiv:1703.01340 (2017).
[87] Jo, S.H., Chang, T., Ebong, I., Bhadviya, B.B., Mazumder, P. and Lu, W.: Nanoscale memristor device as synapse in neuromorphic systems, *Nano Letters*, Vol.10, No.4, pp.1297–1301 (2010).
[88] Xia, L., Liu, M., Ning, X., Chakrabarty, K. and Wang, Y.: Fault-tolerant training with on-line fault detection for rram-based neural computing systems, *ACM/EDAC/IEEE Design Automation Conference* (*DAC*), pp.1–6 (2017).
[89] Wang, Y., Wen, W., Liu, B., Chiarulli, D. and Li, H.: Group scissor: Scaling neuromorphic computing design to large neural networks, *ACM/EDAC/IEEE Design Automation Conference* (*DAC*), pp.1–6 (2017).
[90] Shelby, R.M., Burr, G.W., Boybat, I. and Di Nolfo, C.: Non-volatile memory as hardware synapse in neuromorphic computing: A first look at reliability issues, *IEEE International Reliability Physics Symposium* (*IRPS*), pp.6A.1.1–6A.1.6, IEEE (2015).
[91] Gong, N., Idé, T., Kim, S., Boybat, I., Sebastian, A., Narayanan, V.

and Ando, T.: Signal and noise extraction from analog memory elements for neuromorphic computing, *Nature Communications*, Vol.9, No.1, p.2102 (2018).
[92] Alibart, F., Gao, L., Hoskins, B.D. and Strukov, D.B.: High precision tuning of state for memristive devices by adaptable variation-tolerant algorithm, *Nanotechnology*, Vol.23, No.7, p.075201 (2012).
[93] Song, C., Liu, B., Wen, W., Li, H. and Chen, Y.: A quantization-aware regularized learning method in multilevel memristor-based neuromorphic computing system, *IEEE Non-Volatile Memory Systems and Applications Symposium* (*NVMSA*), pp.1–6 (2017).
[94] Courbariaux, M., Hubara, I., Soudry, D., El-Yaniv, R. and Bengio, Y.: Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or −1, arXiv preprint arXiv:1602.02830 (2016).
[95] Van Pham, K., Van Nguyen, T., Tran, S.B., Nam, H., Lee, M.J., Choi, B.J., Truong, S.N. and Min, K.-S.: Memristor Binarized Neural Networks, *Journal of Semiconductor Technology and Science* (*JSTS*), Vol.18, No.5, pp.568–577 (2018).
[96] Nandakumar, S., Le Gallo, M., Boybat, I., Rajendran, B., Sebastian, A. and Eleftheriou, E.: Mixed-precision architecture based on computational memory for training deep neural networks, *IEEE International Symposium on Circuits and Systems* (*ISCAS*), pp.1–5 (2018).
[97] McGrath, D.: Intel, Numonyx claim phase-change memory milestone, *EE Times* (2009).
[98] Mishra, A.K., Dong, X., Sun, G., Xie, Y., Vijaykrishnan, N. and Das, C.R.: Architecting on-chip interconnects for stacked 3D STT-RAM caches in CMPs, *ACM SIGARCH Computer Architecture News*, Vol.39, No.3, pp.69–80 (2011).
[99] Kawahara, A., Azuma, R., Ikeda, Y., Kawai, K., Katoh, Y., Hayakawa, Y., Tsuji, K., Yoneda, S., Himeno, A., Shimakawa, K., et al.: An 8 Mb multi-layered cross-point ReRAM macro with 443 MB/s write throughput, *IEEE Journal of Solid-State Circuits* (*JSSC*), Vol.48, No.1, pp.178–185 (2013).
[100] Liu, T., Yan, T.H., Scheuerlein, R., Chen, Y., Lee, J.K., Balakrishnan, G., Yee, G., Zhang, H., Yap, A., Ouyang, J., Sasaki, T., Addepalli, S., Al-Shamma, A., Chen, C., Gupta, M., Hilton, G., Joshi, S., Kathuria, A., Lai, V., Masiwal, D., Matsumoto, M., Nigam, A., Pai, A., Pakhale, J., Siau, C.H., Wu, X., Yin, R., Peng, L., Kang, J.Y., Huynh, S., Wang, H., Nagel, N., Tanaka, Y., Higashitani, M., Minvielle, T., Gorla, C., Tsukamoto, T., Yamaguchi, T., Okajima, M., Okamura, T., Takase, S., Hara, T., Inoue, H., Fasoli, L., Mofidi, M., Shrivastava, R. and Quader, K.: A 130.7mm^2 2-layer 32Gb ReRAM memory device in 24nm technology, *IEEE International Solid-State Circuits Conference Digest of Technical Papers* (*ISSCC*), pp.210–211 (2013).
[101] Adam, G.C., Hoskins, B.D., Prezioso, M., Merrikh-Bayat, F., Chakrabarti, B. and Strukov, D.B.: 3-D memristor crossbars for analog and neuromorphic computing applications, *IEEE Trans. Electron Devices* (*T-ED*), Vol.64, No.1, pp.312–318 (2017).



Chaofei Yang received B.S. from Tsinghua University and M.S. from University of Pittsburgh in 2017. He is a Ph.D. student in the Electrical and Computer Engineering department at Duke University. His research interests include Deep Learning Security and Neuromorphic Computing.

Ximing Qiao obtained his B.S. in computer science at Peking University, Beijing, China, and is currently a Ph.D. student in computer engineering at Duke University. His research interest lies at the junction of machine learning and computer architecture, focusing on fast and efficient machine learning algorithm and hardware.

Yiran Chen received B.S. and M.S. from Tsinghua University and Ph.D. from Purdue University in 2005. After five years in industry, he joined University of Pittsburgh in 2010 as Assistant Professor and then promoted to Associate Professor with tenure in 2014, held Bicentennial Alumni Faculty Fellow. He now is the Professor of the Department of Electrical and Computer Engineering at Duke University and serving as the director of NSF Industry-University Cooperative Research Center (IUCRC) for Alternative Sustainable and Intelligent Computing (ASIC) and co-director of Duke Center for Evolutionary Intelligence (CEI), focusing on the research of new memory and storage systems, machine learning and neuromorphic computing, and mobile computing systems. Dr. Chen has published one book and more than 350 technical publications and has been granted 93 US patents. He serves or served the associate editor of several IEEE and ACM transactions/journals and served on the technical and organization committees of more than 50 international conferences. He received 6 best paper awards and 13 best paper nominations from international conferences. He is the recipient of NSF CAREER award and ACM SIGDA outstanding new faculty award. He is the Fellow of IEEE and Distinguished Member of ACM, a distinguished lecturer of IEEE CEDA, and the recipient of the Humboldt Research Fellowship for Experienced Researchers.

(Invited by Editor-in-Chief: *Nozomu Togawa*)