Recent Progress on Memristive Convolutional Neural Networks for Edge Intelligence

Yi-Fan Qin, Han Bao, Feng Wang, Jia Chen, Yi Li,* and Xiang-Shui Miao*

1. Introduction

Recently, due to the development of big data and computer technology, artificial intelligence (AI) has received extensive attention and made great progress. Edge intelligence pushes the computing center of AI from the cloud to individual users, making AI closer to life, but at the same time puts forward higher requirements for the realization of hardware, especially for edge acceleration. Taking convolutional neural networks (CNNs) as an example, which show excellent problem-solving capabilities in different fields of academia and industry, it still faces issues of enormous computing volume and complex mapping architecture. Based on the computing-in-memory property and parallel multiply accumulate (MAC) operations of the emerging nonvolatile memristor arrays, herein the recent research progress of the edge intelligence memristive convolution accelerator is summarized. Furthermore, aiming at improving memristive convolutional accelerators, two potential optimization schemes are also discussed: The compression methods represented by quantization show great potential for static image processing, and the combination of a CNN with a long short-term memory (LSTM) neural network makes up for the CNN’s shortcomings of dynamic target processing. Finally, the future challenges and opportunities of edge intelligence accelerators based on memristor arrays are also discussed.

However, the limited bandwidth and computing resources of the traditional computer system greatly restrict the running speed when faced with the increasing scale of deep neural networks (DNNs). The traditional von Neumann architecture separates data storage and computing. Frequent and inefficient movement of data between the processor and memory or off-chip storage brings latency and energy consumption issues, while the mismatch between data transmission and data processing becomes a bottleneck in the implementation of deep learning in hardware. Due to the high-bandwidth and high-parallelism requirements of deep learning, data-intensive artificial intelligence (AI) applications have been dominated by cloud computing; that is, edge devices act as data-collecting interfaces and pass data to clustered cloud computer centers for computing to achieve deep learning.[6] Such AI applications place high requirements on network bandwidth and latency, and take the privacy leakage issue to users.[5] For example, in areas with poor network conditions, Tesla’s AI autonomous driving will become unreliable and even life-threatening. With the popularization of deep learning, the efficient AI applications that can be seen daily are becoming an urgent need. Edge intelligence is a concept relative to cloud intelligence.[6] Edge computing requires real-time intelligence on devices with strict budgets for energy consumption and device area, such as smart watches and drones. It pushes cloud services from the network core to the edge of the network that is closer to Internet-of-things (IoT) devices and data sources, and then builds up an end-to-end network. Physical proximity to the information-generation sources is the most crucial characteristic emphasized by edge computing, wherefore high energy efficiency, small size, low latency, and high privacy protection become valued characteristics for edge intelligence.[7–9]

With the combination of hardware and AI, devices dedicated for deep learning have emerged. These devices are called neural network accelerators. The combination of a traditional complementary metal–oxide semiconductor (CMOS) and emerging nonvolatile memory provides a considerable wealth of possibilities for AI accelerators.[10–13] The use of memory technology as a synaptic weight matrix storage unit has set a foundation for the hardware implementation of neuromorphic computing systems. In some prominent AI chips, traditional memories have been utilized; for example,
static random-access memories (SRAMs) are used in IBM's TrueNorth and Tsinghua's Tianjic and Thinker chips, and embedded dynamic random access memory (DRAM) is used in Dadiannao architecture.[14-17] A SRAM cell is a bistable transistor structure typically made of two CMOS inverters connected back to back. Due to the low field-effect transistor (FET) barrier height (0.5 eV), the charge constantly needs to be replenished from an external source and therefore SRAM always needs to be connected to a power supply.[18] Low operating power, high operating speed, and high reliability help SRAM to be applied to neuromorphic edge computing. Recently, a 16 kb 65 nm SRAM-based chip showed 139 TOPS/W efficiency and 98.3% accuracy of Mixed National Institute of Standards and Technology (MNIST) dataset and 85.7% of Canadian Institute for Advanced Research (CIFAR-10) dataset, and another work shows 658 TOPS/W efficiency for 98.6% of MNIST.[19,20] However, its low-storage-density and high-bandwidth requirements have become its main obstacles to edge intelligence.[21] A DRAM cell consists of a capacitor placed in series with an FET and necessitates periodic refresh.[18] High memory density is a part of DRAM’s advantages. However, with the increase in the number of synaptic weights and the increase in parallelism requirements, DRAM has significantly reduced the density to increase the random-access speed. Here, the von Neumann bottleneck appears. For instance, the fourth-generation double data rate (DDR4) synchronous dynamic random-access memory (SDRAM) chip offers memory density of up to 32 Gb with data rate (DDR4) synchronous dynamic random-access memory.

Table 1 shows a comparison of some representative works on chip-level demonstrations and design of hardware neural network solutions. Since the connection between the physical TiO2 resistive switching device and the 40-year-old concept was established, the memristor has been extensively studied and is now regarded as the promising device for neuromorphic computing.

| Chip          | Memory   | Technology [nm] | Area [mm²] | Energy efficiency          | Power consumption       | Accurancy [%]       | Network type  |
|---------------|----------|-----------------|------------|-----------------------------|-------------------------|---------------------|---------------|
| TrueNorth     | SRAM     | 28              | 430        | 400 GSOPS W⁻¹              | 65–300 mW               | 92.34 (MNIST)       | SNN           |
| Tianjic       | SRAM     | 28              | 14.44      | 469 GSOPS W⁻¹              | 950 mW                  | N.A.                | ANN/ SNN      |
| Reference     | SRAM     | 65              | 12.6       | 658 TOPS W⁻¹               | 0.8 μJ/pattern (MNIST)  | 98.6 (MNIST)        | CNN           |
|               |          |                 |            |                             | 3.55 μJ/pattern (CIFAR-10) |                     | 83.3 (CIFAR-10) |
| Eyeriss       | DRAM     | 65              | 16         | 246 GOPS W⁻¹               | 200–300 mW              | 80.2 (top five for ImageNet) | CNN          |
| Thinker-II    | DRAM     | 28              | 6          | 19.9 TOPS W⁻¹              | 10.2 mW                 | N.A.                | CNN           |
| Reference     | NOR flash| 180             | <1         | N.A.                        | <20 μJ/pattern          | 94.65 (MNIST)       | ANN           |
| Reference     | STT-MRAM | 22              | 3          | N.A.                        | 0.23 pJ/bit⁻¹           | N.A.                | N.A.          |
| IBM (PCM)     | PCM      | 90              | N.A.       | 28065 GOPS W⁻¹             | 53.87 mW                | 97.94 (MNIST)       | ANN           |
|               |          |                 |            |                             |                         |                     | 88.29 (CIFAR-10) |
| Reference     | Memristor| 180             | N.A.       | 187.62 GOPS W⁻¹            | 306.7 mW                | 94.6 (breast cancer screening dataset) | ANN |
| Reference     | Memristor| 130             | 0.0704     | 11014 GOPS W⁻¹             | 7.438 mW                | 96.2 (MNIST)        | CNN           |
| Reference     | Memristor| 65              | N.A.       | 16.95 TOPS W⁻¹             | N.A.                    | 98.8 (MNIST)        | CNN           |
as a high-potential solution for information storage and neuromorphic computing.\cite{18,38,41} Although memristive resistive switching behaviors have already been observed in various materials, such as binary metal oxides, perovskites, chalcogenides, organic materials, and low-dimensional materials, the most well-balanced device performance could be obtained in transition metal oxide–based devices, for instance, HfO$_2$, TaO$_x$, and Al$_2$O$_3$, which are fully compatible to the CMOS process with low cost.\cite{42,46} The main switching mechanism could be ascribed to the redox reactions and ion migration, driven by the electric field, chemical potential, and temperature gradients.\cite{47,48} The key performance metrics include device scalability, integration density, operation speed, power consumption, and multilevel states. Pi et al. fabricated Pt/TiO$_x$/HfO$_2$/Pt memristor crossbar arrays with a 2 nm feature size and a single-layer density up to 4.5 Tb in.$^{-2}$, comparable to the information density achieved in state-of-the-art 64-layer and multilevel 3D-NAND flash.\cite{49} Up to 8-layer 3D vertical integration has been experimentally demonstrated by a bit cost scalable structure and a nonorthogonal alignment architecture, toward the ultimate (4F$^2$/n) scalability.\cite{50,51} The operating speed of the memristors has reached a few nanoseconds, with a record speed of 85 ps.\cite{52,54} Such a speed is comparable to DRAM and other transistor-based devices, and much faster than the operation speed of the industrial mass-produced flash.\cite{55,56}

The memristor is one of the high-potential device solutions as it allows for the implementation of analog synaptic weight and analog computing in an energy-efficient manner. On one hand, the gradual tuning of the memristor resistance emulates the modification of the synaptic weight, which can be guided by the important local learning rule called spike-timing-dependent plasticity (STDP).\cite{57} This feature provides a feasible route for the construction of hardware spiking neural networks (SNNs).\cite{58} On the other hand, when the memristors are organized in a crossbar array, vector-matrix multiplication (VMM) operation can be executed by exploiting the physical rules with O(1) computational complexity.\cite{59} This efficient analog computing has been demonstrated to accelerate various computation-heavy algorithms for deep learning and scientific computing.\cite{59,61}

Perhaps most importantly, a memristive array can not only store the weights of the neural network stably, but also realize matrix multiplication and addition in parallel using physical laws. Through this so-called in situ in-memory computing method, we can envisage a disruptive revolution over the traditional computer system with von Neumann architecture. To date, compared with an advanced graphic processing unit (GPU), a recently reported memristive AI accelerator shows more than two orders of magnitude better energy efficiency and one order of magnitude better performance density.\cite{60}

In this article, we want to present a comprehensive review on the recent progress of specific AI accelerators, the memristive CNN accelerators for edge intelligence applications, from the perspectives of network topologies, data processing methods, and memristor performances. This article is organized as follows. Section 1 introduces the background of edge intelligence and memristive in-memory computing. Section 2 introduces the working principle of memristive CNN accelerators and the recent hardware implementation progress. Section 3 summarizes the edge-adapted CNN accelerators, especially for static image processing. Section 4 focuses on convolutional long short-term memory (LSTM) neural networks aiming at dynamic pattern processing. Section 5 summarizes this article, provides our prospect of memristors and accelerators for edge intelligence, and foresees future opportunities and challenges.

2. Memristive Convolutional Accelerator

In recent years, under the joint inspiration of biological neuroscience and computer science, artificial neural networks (ANNs) have achieved success that is comparable to or even better than humans in fields such as image recognition.\cite{62} ANNs based on field-programmable logic gates (FPGAs) and general-purpose GPUs have demonstrated certain flexibility and speed advantages, but mixed-signal architectures based on in-memory computing are still researched, focusing on energy efficiency and area-saving edge intelligence.\cite{18}

The three typical ANNs are the multilayer perceptron (MLP), CNN, and SNN. The MLP is also known as the first-generation ANN and relies on stochastic gradient descent and error backpropagation (BP). Many memristive MLP networks have been published.\cite{63,64} Prezioso et al. classified the 3 × 3 binary image well by a one-layer MLP, which was implemented by an Al$_2$O$_3$/TiO$_{2.3}$ memristor crossbar.\cite{65} Li et al. reported an experimental demonstration of in situ learning in a multilayer neural network implemented in a 128 × 64 memristor array.\cite{66} The MLP does show good functions in areas such as pattern recognition, but when faced with more complex tasks and data sets, its rapid expansion has brought difficulties in hardware implementation. In other words, compared with CNNs and other latecomers that are more biased toward a specific function and domain network, the original MLP has no advantages in terms of energy consumption and task complexity. The SNN is more biologically plausible and exhibits favorable properties such as low power consumption, fast inference, and event-driven information processing. The spiking neurons compute with sparse and asynchronous spikes that are temporally precise,\cite{18} whereas the network is trained with local learning rules, such as STDP, which allows detecting spatiotemporal patterns as features. The STDP adjusts the synaptic weight based on the relative time between the output and input neuron spikes, directly simulating the activity of nerve cells and achieving biological functions in electronic devices. So far, STDP functions have been successfully implemented, using nonvolatile memories, and then serve as the learning rules for hardware SNNs.\cite{67,68,69,70} However, various SNNs still cannot achieve the accuracy of the traditional ANNs trained with BP. Other restrictions include the lack of compact bioplausible spiking neurons and appropriate training algorithms to efficiently take advantage of spiking information processing and the absence of spike-based benchmark datasets and metrics to evaluate the real-world performance of SNNs. Therefore, this hardware-efficient network is still in its infancy and needs further intensive investigation. In summary, the CNN is one of the mainstream choices for memristive neuromorphic computing and edge computing.

2.1. Introduction of the CNN

The convolutional neural network is one of the typical DNNs. Its predecessor, the neocognitron, was proposed by the idea of
imitating the visual cortex, where different types of visual cells can be affected by the features of visual stimuli in their receptive field.\cite{72,73} By introducing an error-backpropagation algorithm, LeCun proposed the first CNN model capable of recognizing hand-written numbers.\cite{74} Since then, many advanced CNN models, such as LeNet, AlexNet, VGG, GoogLeNet, and ResNet, have been proposed, and these models showed great performance in pattern recognition when compared with the traditional shallow MLP.\cite{1,3,75–77}

Convolution is a very important operation in mathematics. It denotes the overlap of the input function \( f \) as it is shifted over the other input function \( g \), written as \( f \ast g \)

\[
f(t) \ast g(t) = \int_{-\infty}^{+\infty} f(\tau)g(t - \tau)d\tau
\]

Convolution operations in the CNN are basically sliding the weight kernel throughout the 2D images, as described in Figure 1a. Thus, the number of trained parameters and memory cost are considerably reduced, and this operation is called the weight-sharing method. And by importing images without flattening, the CNN can retain the original features and remain robust to transformations such as flipping, rotation, and translation. Moreover, trained CNNs are capable of transferring between different tasks and datasets with the ability of extracting basic features while disregarding specific tasks. This is known as the fine-tuning method, which is appealing for model extension and reuse.\cite{78}

CNNs are usually composed of three types of layers—convolutional layers, pooling layers, and fully connected layers—as shown in Figure 1b. In convolutional layers, input feature maps are convolved through planes and channels to extract different image features. The abstraction of features is strengthened through convolutional layers. Pooling layers help the network downsample features, decrease the number of parameters, and increase the robustness of the network. They include max pooling kernels and average pooling kernels and often come after convolutional layers. Fully connected layers usually serve as the final layers in a CNN, representing the classifier of features.

2.2. Memristor-Based CNNs

The remarkable advantage of mapping CNN weights into memristor arrays by utilizing the connected topology is performing MAC operations parallelly and efficiently. With the corresponding memristor conductance as the weights, voltage signal as the input, and current as the output, MAC operations can be directly preformed in memory to reduce data movement according to Ohm’s law and Kirchhoff’s current law. Thus, energy and latency can be reduced significantly. However, to implement the hardware of memristive CNNs, it is essential that the following hierarchical problems are solved. 1) Weights mapped to memristors will suffer from device nonideal characteristics, leading to the reduction of network accuracy. 2) The weight-sharing and sparsely connected nature of convolutional operation require extra weight and data mapping design for a memristor crossbar array compared with fully connected layers. 3) Fully hardware implemented deep CNNs require large device array fabrication, complex peripheral circuit design, and optimization of array non-ideal factors.

As for the weight update nonlinearity and asymmetry that have plagued researchers, there are now many solutions to alleviate the adverse effects of these nonideal factors in AI systems, such as quantization and the modulation of voltage pulse amplitude or width.\cite{79–83} Regarding the level of memristive arrays, the statistical characteristics of memristors such as intrinsic variation of cycle to cycle (C2C) and device to device (D2D) are particularly important. At the same time, large-scale arrays will also bring challenges to peripheral circuits and integration technologies, such as IR drop. Optimizations for three levels of material, device, and array are urgent and necessary.\cite{13} Numerous attempts have led memristors to explore new areas in the field of brain-inspired intelligence.

In this section, we will present some important works that have solved or made a critical further step on those essential problems to build a sketch on implementing memristor array-based hardware CNNs.

Memristors are the basic components in memristive hardware neural networks. Depending on the number of conductance states, they can be generally divided into two types, analog memristors and binary memristors. In ideal situations such as software simulation, weights are precisely tuned based on error functions calculated from BP in floating point. However, for actual analog memristor devices, they inevitably suffer from various nonideal factors with different effects on network accuracy according to the consensus of recent studies.\cite{84–88} Generally speaking, the influence of D2D variations is usually trivial to accuracy, which can be concealed by training. The impact of
C2C variations is critical and can be mitigated by device replication. Moderate stochasticity in device conductance is proven to improve network training accuracy by avoiding the local minimal and overfitting. Device write asymmetry and nonlinearity jointly affect accuracy during weight update; more precisely, asymmetric write nonlinearity is more critical than symmetric write nonlinearity. That can be eliminated by methods such as reading conductance before update, verifying conductance after tuning, and updating voltage ramping at the cost of peripheral circuit design and extra time consumption. Limited conductance states of analog memristors can be implemented as quantized weights with a truncated weight range to reduce the effect. Endurance and retention are also important characteristics to ensure network accuracy through time. For binary memristors that only have two states, they can be used in binary/ternary neuron networks, or grouped into a set to simulate quantized analog devices at the cost of area. The major advantages of binary memristors are their high endurance and retention of each conductance state compared to analog ones, and it is beneficial for inference-only hardware neuron networks.

To achieve parallel processing of convolution operation in the memristor crossbar array, Gao et al. first experimentally demonstrated their method in 2016, as shown in Figure 2a.[91] 2D kernel matrices were dimensionally reduced into 1D column vectors. Then the weight column vectors were mapped as the difference of two cells in the adjacent columns in their fabricated 12 × 12 memristor crossbar array, to achieve both positive and negative weights. And the partial input images with each receptive field during kernel sliding were programmed into sequential 1D column vectors and conducted into row lines of the array. By such a dimension reduction method, convolution operations could be mapped as VMM and parallelly processed by the memristor array like those in fully connected layers. Prewitt kernels were used to detect horizontal and vertical edges of the MNIST handwritten digits as a proof-of-concept demonstration. Li et al. presented another outstanding work by extending the demonstrated memristor array size to as large as 128 × 64 one-transistor one-resistor (1T1R) analogue cells for the first time.[92] In their work, completely analogue VMMs (analogue-input-vector and analogue-weight-matrix multiplication) had adequate accuracy for signal processing, image compression, and convolutional filtering. Other weight mapping strategies such as mapping into sparse matrices or mapping into spatially localized groups were also discussed in simulations aiming to improve parallelism, thus reducing data storage requirement as well as counterpart latency and energy.[93,94]

Figure 2. a) Schematic of parallely processing convolution operation in the memristor crossbar array by dimensional reduction of a kernel matrix. Reproduced with permission.[91] Copyright 2014, IEEE. b) Photograph of the integrated hardware subsystem (left) and processing element (PE) chip (right). c) Schematic of the demonstrated CNN structure and corresponding weight mapping design into PEs. Reproduced with permission.[36] Copyright 2020, IEEE.
In a recent breakthrough, Yao et al. proposed a fully hardware-implemented memristor CNN in experimentally fabricated and integrated eight 2048-cell 1T1R analog memristor arrays, as shown in Figure 2b,c.[36] The weights of two convolutional layers and one FC layer were well trained on software and transferred into arrays. For forward propagation and inference, inputs were encoded as a series of identical pulses according to their quantized value, which eliminated input analog to digital converters (ADCs). In this process, outputs were shifted and added to present the true output value. Max pooling function and rectified linear unit (ReLU) activation functions were realized by a peripheral ARM core. The freshly transferred memristor CNN suffered a relatively high accuracy loss because of device nonideal characteristics and weight-transfer error. To mitigate those effects, weights of the fully connected layer were calculated in software and then updated in hardware to achieve both efficiency and partial recovery of accuracy, which is called the hybrid training method. Moreover, the approach of transferring weights of the same kernel into three independent subarrays was applied to process three data batches at the same time for further acceleration. Their investigation provided a comprehensive approach of a fully hardware-implemented CNN based on a 1T1R analog memristor array and put forward extendable methods such as hybrid training and parallel computing on multiple arrays. Benchmark metrics of a single macro core with 8 bit input showed 11 041 GOP s⁻¹ W energy efficiency and 1164 GOP s⁻¹ mm² performance density, with more than 96% accuracy for the MNIST dataset.

Comprehensive demonstrations of a binary memristor-implemented CNN were proposed on a foundry-fabricated 1 Mb 1T1R single-level (binary) cell memristor array by Chen et al., as shown in Figure 3a,b.[37] Ternary weights of convolutional kernels were mapped into the difference of two subarrays (+1, 0, and −1), while inputs were encoded as digital signals (0 and +1). Taking the positive weight subarray, for example, with a predetermined number of parallel inputs \( N \), the binary inputs and binary weights allow the outputs of MAC to be simply deduced by comparing with programmed \( I_{\text{reference}} \), where \( I_{\text{reference}} = xI_{\text{LRS}} + (N-x)I_{\text{HRS}} \) (LRS and HRS refer to low resistance state and high resistance state, respectively). With this approach, the high area overhead, high power consumption, and large latency caused by high-precision ADCs used in analogue arrays could be eliminated. A field programmable gate array (FPGA) was used as the controller and digital processing for activation and pooling. Logic operations such as XOR in the same array were also proposed. The nonvolatile computing-in-memory (nvCIM) macro performs three-input Boolean
logic operations by 4.9 ns access time and MAC operations by 14.8 ns computing time and reduces the MAC energy efficiency to 16.95 TOPS W\(^{-1}\) with just 0.97% inference error rate increase. The peripheral circuit design discussed by Xue et al. from the same team further optimized their strategy.\(^{[95]}\) The circuit design of a weighted current translator was proposed by using different sizes of transistors for multitbit dot product in MAC. A positive–negative current-subtractor circuit design was applied to reduce the total output current. Multiple camping transistors with a voltage detector and a digital controller were adopted to clamp read voltage from a wide current range. A 3 bit current-mode sense amplifier with small input offset was proposed to tolerate small read margin from quantified MAC output. All their designs were in the form of a sequential circuit, which is beneficial for the integration of a memristive CNN and traditional digital circuits.

Normal passive memristor crossbar arrays usually suffer from a dilemma between array size and sneak path current, which prevents them from being used to achieve DNNs with low device cost.\(^{[97,98]}\) On the other hand, 2D planar arrays have limited device density and simplified connections compared to 3D arrays, which means 2D arrays are not beneficial to implement the complex topology of DNNs. Therefore, a very recent work by Lin et al. has expanded a hardware-implemented CNN into a tailored eight-layer 3D memristor array, as shown in Figure 3c, d, to provide a high degree of functional complexity with a relatively negligible array size effect.\(^{[96]}\) In their specially designed array structure, memristor devices were stacked into row bank structures between the input and output electrodes. Thus, the entire array was partially connected at each row bank, resulting in a large array size with small sneak path current, high sensing accuracy, and high voltage delivery efficiency compared with traditionally designed 2D, 3D memristor arrays under similar circumstances. Moreover, the row bank structure introduced a spatial overlapping but electrical isolating characteristic between vertical devices. The structure enables parallel convolution operations with input pixels shifted and partially replicated, as well as weight kernels replicated to multiple device sets, rather than a full rearrangement of both input pixels and kernel weights, which simplified the design of cascade and peripheral circuit overhaul. Although their partial connected memristor array is not beneficial to implement fully connected layers, which have a lot more parameters, such a tailored 3D memristor array design is still a critical avenue toward a convolutional accelerator for high density near sensor and edge intelligence applications.

Up to now, most memristive neural hardware networks are designed as inference-only accelerators, in which the weights are previously trained on the software and then transferred to the memristor arrays. By alleviating the learning process on the array, counterpart circuits and energy can be eliminated, and the requirements of memristor endurance can be liberalized with retention becoming critical. Nevertheless, many simulation studies focusing on on-chip learning design are still worth noting. In these studies, error signals are obtained by importing activations into the weight array reversely.\(^{[99,100]}\) Such a method can improve the flexibility of the network, but the complexity increases, so it is more suitable for unsupervised learning applications.

In short, memristor-based CNNs have made impressive progress in recent years, showing great potential for hardware-implemented accelerators, but further optimization toward edge intelligence applications is still desirable.

3. Edge-Adapted CNNs

A CNN is competent for different scopes of deep learning tasks, including CIFAR-10 and ImageNet, which demand huge energy consumption and memory space. Generally, the large model and deep structure promote the network performance, but deeper networks demand huge energy consumption and memory space, and this is contrary to the trend toward low-power, portable, and time-saving work scenes. Thus, the bulk of CNNs could be performed with compressed in-memory computing for better adaptation of edge devices. For this issue, researchers mainly focus on two major solutions: One is to reduce the size of the network by means of compression and encoding, such as the quantization network, and another is to design a more effective network architecture and achieve acceptable accuracy with a relatively small model size, such as MobileNet and SequeezeNet.\(^{[101,102]}\)

3.1. CNN Quantization

Quantization (or low precision) is one of the most widely used compression methods. As shown in Figure 4, the continuous values are quantized, i.e., discretized to several specific values, which means the weights or activations are represented by correlated discrete values. Data precision is generally ≥8 bit (fixed point) for inference.\(^{[103,104]}\) Nvidia has reported that the inference accuracy maintains when the 8 bit TensorRT works on AlexNet, VGG, and ResNet models.\(^{[104]}\) Recent studies are devoted to weighing the trade-off of acceptable accuracy loss and network efficiency, and these low-precision compressed CNNs better cater to the needs of edge devices for their small size and high performance. For example, Han et al. combine pruning, trained quantization and Huffman coding to reduce the storage required by AlexNet by 35× and VGG-16 by 49× without loss of accuracy on the ImageNet dataset.\(^{[105]}\)

3.1.1. Memristive Binary CNN

A binary neural network (BNN) quantizes the parameters to binary values (1 bit) and modifies the BP process to adapt the quantization. Courbariaux et al. introduced a method to train a binarized network with binary weights and activations at run-time. They calculated the gradient of the binary parameters

**Figure 4. Schematics of quantization.** Continuous analog values (real values) are divided into many equal or unequal intervals by algorithms, and the analog value in each interval is replaced by a fixed value (quantized values).
and updated real values at train-time.\textsuperscript{[106]} The weights and activations are quantized by Equation (2). Thus, the number of convolutional kernels is significantly reduced in a binary convolutional neural network (BCNN), which in turn increases the convolutional kernel reuse rate and reduces time complexity by 60%. As a special BNN, the XNOR-Net consisted of four stages: batch normalizations, binary activation, XNOR convolution, and pooling.\textsuperscript{[107]} The network introduced scale factors to binarized weights and activations, minimized the $L_2$ norm of the parameters after binarization and the original parameters to further improve the accuracy of the BCNN. Moreover, the feature maps were also binarized. In the XNOR-Net, the analog high-precision MAC operations were replaced by XNOR logic and pop-counting. This significantly improves the speed and efficiency of these operations and provides a promising solution to overcome the device nonideal limitations in the memristive CNN accelerator design.

$$x_b = \text{sign}(x) = \begin{cases} -1, & \text{if } x \leq 0 \\ +1, & \text{if } x > 0 \end{cases}$$ (2)

The memristive binary convolutional neural network (MB-CNN) is an in-memory computing accelerator that achieves convolution computing within memristive crossbar arrays. Tang et al. proposed a prototype for the BCNN forward process and discussed the matrix splitting problem and the pipeline implementation.\textsuperscript{[108]} The memristor accelerator saved 58.2% energy consumption and 56.8% storage area compared with the multibit CNN structure for AlexNet on ImageNet. Considering the current array fabrication technology, a pipeline strategy was applied to introduce intermediate data buffering, as shown in Figure 5a. In addition, the BCNN achieved a 0.75% error rate on a 3 bit memristor taking device variation into account, and the device variation introduced a less than 0.01%

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure5.png}
\caption{a) Overall structure of the MB-CNN accelerator. Reproduced with permission.\textsuperscript{[108]} Copyright 2017, IEEE. b) The customized bit-cell design for XNOR implementation. Reproduced with permission.\textsuperscript{[109]} Copyright 2018, IEEE. c) Every synapse in a DX array contains two memristors with opposite states. Reproduced with permission.\textsuperscript{[110]} Copyright 2019, IEEE.}
\end{figure}
error rate increase in case of the 3 bit (or larger bit-level) memristor, which got worse in full-bit-level mode. Interestingly, the energy and area consumption got lower because of the reuse of convolution kernels.

With both weights and activations binarized to $+1$ or $-1$, the high-precision MAC operations can be replaced by XNOR logic and bit-counting operations.\cite{106} The two-transistor-two-resistor structure is a mature solution to implement a MB-CNN. Sun et al. conceived of a resistive random access memory (RRAM) synaptic architecture (XNOR-RRAM) with a bit-cell design of complementary word lines that implemented equivalent XNOR and bit-counting operations in a parallel fashion.\cite{109}

The $+1/-1$ weights were represented by a pair of IT1R devices in which the memristors kept opposite conductance states (LRS/HRS) as shown in Figure 5b. The system with $128 \times 128$ subarray size and 3 bit multilevel sense amplifiers could achieve an accuracy of 86.08% for a CNN on CIFAR-10, showing 2.39% degradation compared to that of ideal BNN algorithms. The simulated energy efficiency of 141.18 TOPS W$^{-1}$ improved $\approx 33 \times$ compared to the conventional sequential row-by-row read-out memristive architecture. In addition, the discussion on data transfer offset between memristive arrays provided valuable reference at current small array size fabrication technology.

A new approach based on a differential crosspoint (DX) memristor array for enabling parallel MAC operations was reported to significantly reduce computational complexity and relaxed memory requirements in an MB-CNN.\cite{110} One synapse was composed of two opposite-resistance-state memristors, as shown in Figure 5c. Each DX unit contained a $64 \times 64$ small memristive array and formed the accelerator as a subunit. The subarrays minimized parasitic resistance and capacitance for quicker MAC operations. The binary weights were mapped in DX units and prevented the BNN from weight movement, which improved the kernel reuse rate. Meanwhile, the proposed neighbor shifting scheme reduced the input data traffic by two-thirds and improved the time efficiency. Energy efficiency of 160 TMACs $^{-1}$ W$^{-1}$ was estimated and high robustness was proved by simulation.

Cai et al. demonstrated a nonvolatile MB-CNN accelerator designed for the IoT system and the system had an advantage of static power consumption in standby state.\cite{111} Vieira et al. showed a custom binary dot product engine (BDPE) for accelerating the MB-CNN inference phase, and the BDPE improved performance by 11.3% and achieved 7.4% energy savings.\cite{112} In MB-CNNs, the accelerator has generally shown significant time and energy efficiency. However, the limitation was that the BCNNs counteracted the accuracy degradation by keeping more parameters and an increased number of training epochs.\cite{106} When it comes to hardware implementation, significant concurrent wire resistance would occur to the large memristive array, and the IR drop and offset degraded the BCNN accuracy and efficiency. Thus, utilizing subarrays might be a common current solution for MB-CNNs. In addition, the larger datasets for more complex networks gained unsatisfactory results in BNNs.\cite{106}

That is to say, BCNNs have better performance in handling small tasks on edge devices such as mobile phones and embedded systems.

### 3.1.2. Memristive Multibit CNNs

Compared to BCNNs, the ternary convolutional neural networks (TCNNs) quantize weights to $-1$, 0, and $+1$, which may have stronger expressive abilities than binary precision counterparts and are more effective than the latter.\cite{113-115} Esser et al. demonstrated binary neurons and a ternary weight architecture for a TCNN and proved the feasibility of the TCNN in spiking neurons.\cite{116} The 1 Mb nvCIM macro implemented ternary weight convolution operations based on a IT1R structure and achieved an inference accuracy of 98.8% on the MNIST dataset.\cite{117} Lin et al. accelerated convolution operations by a 3D vertically memristive array with ternary kernel weights and achieved 98.11% accuracy on the MNIST dataset.\cite{96} The new architecture shows the potential to combine low-precision weight with a high degree of functional complexity in memristive devices.

Apart from ternary weight, other low-precision quantization methods are used to simplify and accelerate convolution operations. Cheng et al. mapped a 4 bit quantized deep residual network to an RRAM crossbar, as shown in Figure 6a.\cite{117} The accelerator achieved 88.1% accuracy on the ImageNet benchmark with a loss of 2.5% and showed considerable progress on both computing speed and energy efficiency.

In another study, Yang et al. trained and quantized the CNN at the same time to alleviate the impact of limited parameter precision.\cite{79} Different from direct quantization by mapping parameters to the memristive crossbar after training, the quantization method showed significant accuracy improvement on neuromorphic systems. Based on the study, several suggestions for neuromorphic system designs were proposed. As shown in Figure 6b, Zhu et al. proposed a configurable multiprecision CNN computing framework based on a single-bit RRAM.\cite{118} This framework introduced RRAM-aware quantization to overcome the area and storage burden brought about by binary memristors. The framework reduced 70% computing area and 75% computing energy while significantly reducing memristor requirements.

The performance of a quantization CNN is shown in Table 2.

### 3.2. Optimization of CNN Algorithms and Structures

CNN quantization compresses and reduces the weight size and storage demand, but the convolution still has huge multiplication and addition operations, which restrict the further improvement of convolution efficiency and the promotion of edge applications. Nowadays, the light CNNs which are optimized to compress network size or complexity have become one of the representative ways in the convolution acceleration.

SqueezeNet convoluted feature maps with $1 \times 1$ kernels to compress feature maps and reused the Fire module in the convolution process.\cite{102} MobileNet introduced depth-wise separable convolution to replace the conventional convolution operation and compressed parameter size to accelerate convolution at the same time.\cite{103} Deploying these light CNNs into the memristive hardware is still a challenge that is worth investing in.

Although the optimized CNNs accelerated the computing, the high-precision weights hindered the realization of the network in hardware, and they also put higher requirements for memristive
devices. In addition, the optimization had cut the redundancy of the conventional CNN, which made it hard to combine structure optimization with weight quantization.

### 4. Memristive LSTM Neural Networks

#### 4.1. Basic Principle of LSTM Neural Networks

In general, traditional DNNs can only process static input information, while continuous timing information with different lengths is difficult to process with traditional DNNs, and as such it is difficult to transform dynamic information to static information to be suitable as input. However, a recurrent neural network (RNN) possesses unique abilities to process such information. Each neuron in the hidden layer of an RNN introduces a

---

**Figure 6.** a) In-memory accelerator architecture based on a 4 bit RRAM crossbar for the 4 bit quantized ResNet-50. Reproduced with permission.\[96\] Copyright 2020, Springer Nature. b) Right: the overall architecture design; left: details of the PE, the PE slice, and the joint model. Reproduced with permission.\[118\] Copyright 2019, IEEE.

**Table 2.** Accelerators with quantization CNN.

| Work                  | Quantization approach | Efficiency performance |
|-----------------------|-----------------------|------------------------|
| MB-CNN\[108\]         | W/A: binary           | 13.55 μJ img⁻¹ (LeNet) |
|                       |                       | 2275.34 μJ img⁻¹ (AlexNet) |
| XNOR-RRAM\[109\]      | W/A: binary           | 141.18 TOPS W⁻¹        |
| Crosspoint\[110\]     | W/A: binary/4b        | 160 TMAC s⁻¹ W⁻¹       |
| MB-CNN\[111\]         | W/A: binary           | 366.134 mW             |
| BDPE\[112\]           | W/A: binary           | 0.870 μJ               |
| nvCIM\[37\]           | W/A: ternary/FP16     | 16.95 TOPS W⁻¹         |
| 4 bit CMOS-RRAM accelerator\[117\] | W/A: 4b/4b | 75.2 TOP J⁻¹         |
| Multiprecision CNN framework\[118\] | W/A: multi | 3.44 TOPS W⁻¹       |
recurrent synapse that stores the neuron’s output at moment \( t \) as its own input at moment \( t+1 \), and similarly, the output at moment \( t+1 \) will serve as input at moment \( t+2 \), and so on. In this way, the output of the hidden layer neurons is associated with the state of the previous network and has the ability to process timing information. However, in practice, the error signals in the BP of the network would have a serious problem of gradient vanishing or gradient blowing up when the sequence of input samples is too long. The former problem causes weights to not update or even the network to not work, while the latter causes the weights to oscillate and not converge.

The gradient vanishing problem can be solved nicely by weighted clipping, but there is no proper way to solve the gradient disappearance problem.

Proposed by Hochreiter et al. in 1997, an LSTM neural network is a variant of an RNN. As shown in Figure 7a, LSTM complicates the recurrent neurons of the hidden layer on the basis of RNN, and introduces the variable “cell state” to store the state of a neuron. In addition, extra structures, such as a forgetting gate, input gate, and output gate, have been introduced to modify the cell state at the expense of multiplying the number of trainable synaptic weights of the hidden layer, which it is hoped will solve the gradient vanishing problem. LSTM solves the problem of gradient vanishing in RNNs and has been widely used in machine translation, speech recognition, natural language processing, sentiment analysis, human gait identification, regression prediction, and other fields.

The inference of LSTM can be summarized by the following equations:

\[
a' = \tanh \left( W_a x' + U_a h^{t-1} + b_a \right)
\]

\[
i' = \text{sigmoid} \left( W_i x' + U_i h^{t-1} + b_i \right)
\]

\[
f' = \text{sigmoid} \left( W_f x' + U_f h^{t-1} + b_f \right)
\]

\[
o' = \text{sigmoid} \left( W_o x' + U_o h^{t-1} + b_o \right)
\]

\[
c' = i' \odot a' + f' \odot c^{t-1}
\]

\[
h' = o' \odot \tanh (c')
\]

\[x' \text{ is the output vector of the neurons in the previous layer. } h' \text{ and } h^{t-1} \text{ are the output vectors of this hidden layer of the current time step and the previous time step, respectively. } a', i', f', o' \text{ are the vectors of cell activation, input gate coefficient, forgetting gate coefficient, and output gate coefficient, respectively. } W, U, \text{ and } b \text{ represent the weight matrix of input synapses, recurrent synapses, and bias synapses, respectively. } " \odot " \text{ is the symbol of element-wise multiplication, also known as Hadamard product. The sigmoid function is used to map the coefficients to a positive number less than 1 for filtering the information. For instance, } i' \odot a' \text{ represents the filtering of cell activation by the input gate coefficient; } f' \odot c^{t-1} \text{ represents the filtering of the cell state of the previous time step by the forgetting gate coefficient; } o' \odot c' \text{ represents the filtering of the cell state of this time step by the output gate coefficient and gives the final output } h', \text{ which will be output to the neurons of the next layer and the neurons of this layer in the next time step.}

4.2. Memristive LSTM Neural Networks

As the main process during inference of LSTM remains MAC, it means an LSTM-RNN can also use a memristor crossbar array as an accelerator. Not only that, but its complex internal structure results in a much larger number of trainable parameters. When implementing LSTM on traditional digital hardware, it will face more severe von Neumann bottlenecks and high power and high latency problems than other kinds of neural networks. Therefore, running LSTM on an edge intelligence...
platform with a memristor crossbar array will yield greater benefits than other types of neural networks.

Figure 7b is a typical approach to the weight mapping of LSTM neural networks using a memristor crossbar array. Assuming that each device represents one synapse, we need an array of at least \((m + n + 1) \times 4n\) in size to map an LSTM hidden layer, where \(m\) donates the dimension of the output vector of the previous layer and \(n\) donates the number of neurons in this LSTM hidden layer, in which the four internal neuronal parameters \(a'\), \(i'\), \(f'\), and \(o'\) each occupy \(n\) columns, while input synapses \(W\) occupy \(m\) row lines, recurrent synapses \(U\) occupy \(n\) row lines, and biased synapses \(b\) occupy one row line. To complete a forward propagation process in the LSTM hidden layer, the following actions are required in sequence. Step 1: \(x^t\) and \(h^{t-1}\) are converted to analog voltage signals by digital to analog converters (DACs) and then inputted to the array by lines. Step 2: The matrix–vector multiplication operation is completed by Ohm’s law and Kirchhoff’s current law, and the result is outputted by the column line in the form of current values. Step 3: After the current values are converted by ADCs to digital signals, the sigmoid and tanh functions give the values of \(a', i', f',\) and \(o'.\) Step 4: According to Equation (7) and (8), \(c^t\) is updated and \(h^t\) is outputted. Step 5 comprises keeping \(c^t\) to the next time step while outputting \(h^t\) to the next layer of neurons and this layer of neurons in the next time step.

Yin et al. first proposed a method for training an LSTM neural network on a fully parallel RRAM array in 2018.\textsuperscript{126} They used binary weight and 4-bit activation precision to estimate by simulation that the network had an energy efficiency of \(\approx 27 \text{TOPS} \text{W}^{-1}\) with the Penn Treebank dataset as a benchmark. Due to the complexity of natural language processing tasks, this network is too large to apply on a real memristor array.

Gokmen et al. conducted further simulation work by introducing a resistive processing unit baseline model, with certain nonideal effects, to explore the influence of various external and internal factors on the result of training a memristive LSTM neural network.\textsuperscript{126,130,132} By introducing a stochastic rounding scheme for the input vector instead of rounding to the nearest neighbor scheme, the researchers found that this scheme could reduce the minimum requirement for the resolution of the input vector in an LSTM network from 7 to 5 bit, thereby significantly reducing the complexity and time consumption of peripheral circuits. In addition, the researchers idealized the nonideal parameters, including variation, asymmetry, and conductance states, in the baseline model and concluded that asymmetry has the greatest impact on the network training effect, as shown in Figure 7c. It is worth noting that many of the memristive LSTM simulation efforts, including this study, envision array sizes far beyond the existing real memristor arrays, so how to reduce the array size requirements of LSTM networks to apply them to edge intelligence devices is an urgent challenge.\textsuperscript{126,127,130,133,134}

The first implemented LSTM neural network on a memristor array was reported by Li et al. in 2019.\textsuperscript{126} Using a Ta/HfO\(_2\) memristor to stably store multiple conductance states, the authors implemented the matrix–vector multiplication algorithm in the array. By solving two types of real-world problems, regression and classification, it demonstrated that the memristive LSTM neural network could be the core of a promising edge inference accelerator that has high-energy-efficiency, high-area-efficiency, and low-latency characteristics.

More specifically, a \(128 \times 64\) 1T1R array was fabricated based on a commercial foundry-produced transistor array. A scheme in which two devices represent one weight was implemented by inputting a pair of voltages of equal amplitude and opposite symbols for each of the two devices located on the same column line. Such a weight mapping scheme is more accurate, but also makes the whole network require twice as many devices as the original, i.e., \(2 \times (m + n + 1) \times 4n\). In this work, the write-and-verify approach and the two-pulse scheme were adopted to write and modify the conductance values.\textsuperscript{133} The implementation of these two schemes demonstrates that it is feasible to use weight update schemes already available in other neural networks in memristive LSTM networks.\textsuperscript{92,133,135}

Researchers first verified the feasibility of memristive LSTM with a simple data prediction task in the regression experiment. Then, a human gait recognition task was chosen in the classification experiment. Human gait recognition is an important means of identifying a target when face recognition fails, and as gait recognition requires the extraction of features from a continuous screen for analysis, sending data to the cloud for identification would result in greater bandwidth consumption and higher latency than other biological recognition technologies. Therefore, the study on building LSTM neural networks with edge intelligence devices and doing inference locally is of high application value. As shown in Figure 8a, to adapt the input vectors to the array size, the researchers further downscaled each frame to 50, compressing it to the number of pixel points occupied by the human body profile at each height.\textsuperscript{136} However, even with such an aggressive approach to input sample downscaling, the array has been just barely occupied, and the recognition rate achieved in Figure 8b is not ideal, with a maximum accuracy of 79.1%. It is obvious that the size of the memristor array greatly constrains the application of a memristive LSTM network in practical edge intelligence.

Subsequently, Wang et al. introduced a CNN on the same memristor array and combined it with an LSTM neural network.\textsuperscript{137} Simultaneously utilizing spatial and temporal weight sharing mechanisms, this ConvLSTM maintained equivalence accuracy (\(\approx 92.13\%\)) compared to the fully connected memristive neural network in the MNIST handwritten digit recognition task while cutting the number of synapses in the network by 75% to about only 800 weights.

Figure 8c is an implementation procedure for the ConvLSTM network on a 1T1R array, which is used for MNIST dataset classification. The weights of all convolutional regions in Figure 8c are consistent, achieving spatial weight sharing, while the weights of all time steps are reused, achieving temporal weight sharing. Thus, by performing convolutional operations on the two inputs, both spatial and temporal weight sharing can be realized. Figure 8d provides a clear demonstration of the weight mapping of the ConvLSTM and fully connected layer in a 1T1R array of size \(128 \times 64\).

The aforementioned work successfully implements matrix–vector multiplication operations for LSTM neural networks on memristor arrays and succeeds in reducing the network requirements for array size to an acceptable level by combining a CNN and LSTM. However, operations other than matrix–vector
multiplication still need to be performed in digital devices, which is clearly not in line with our original intention to bypass the von Neumann bottleneck with memristor arrays. Restrictions such as latency and power consumption due to frequent digital-to-analog conversion also constrain its use in edge intelligence. Smagulova et al. proposed CMOS–memristor combined circuits that can be used to perform sigmoid and 
\[ \tanh \]
function operations (Figure 9a) and voltage multiplication operations (Figure 9b).[123]

Wen et al. then proposed a more complete hardware circuit implementation of LSTM. In contrast to the approach shown in Figure 7b, in which the internal parameters of four neurons, \( a_t, i_t, f_t, \) and \( o_t \), are first calculated inside the array, followed by \( h_t \) and \( c_t \) in the peripheral circuit, Figure 9c demonstrates a method that allows the cell output and cell state of each LSTM neuron to be calculated in parallel.[121] Each LSTM unit in Figure 9c contains the structure shown in Figure 9d, where part a, b, and c store the weight matrix of input synapses, recurrent synapses, and bias synapses, respectively, part d implements the sigmoid and \( \tanh \) functions by segmental function approximation, and part e is a voltage multiplier that ultimately yields the output vector and cell state components of this unit. Recently, Han et al. tried to perform element-wise product operations by training the neural network approximator to reduce the additional power consumption and latency problems associated with frequent digital-to-analog conversions and the matching problems between different types of data.[138] All these works have contributed progressively to the implementation of a fully hardware LSTM neural network and its application in edge intelligence devices.

5. Challenges and Outlook

In this review, we present a comprehensive induction on convolution neural networks for memristive implementation. We progressively introduced three levels of CNN memristive accelerators: basic implementation, edge-friendly, and dynamic image processing. We systematically introduce the state-of-the-art convolution accelerators and the impact of memristors, the peripheral circuits on the network, and then explain the quantization methods and implementations for edge intelligent applications. In addition, we show the convolution acceleration of dynamic image processing that is lacking in conventional edge intelligence.

When memristors are applied to neuromorphic and edge computing, the specific needs to be met may be strongly personalized for different tasks. For example, compared to in situ training network accelerators, memristive arrays used only to accelerate the inference process will pose lower device requirements. In fact, deep learning algorithms and memristors are mutually compromising and reinforcing, so it is impossible to give a unified and very exact standard, but some suggestions and requirements can be given tentatively.
For most memristive deep learning accelerators, low-resistance-state devices are usually required to be resistive enough for reducing the on-state current, thereby limiting the IR drop during read and write operation in arrays. At the same time, the write current should be less than 100 nA to reduce the whole chip power consumption. In addition, the read noise standard deviation should be less than 5% of the weight range, and the write noise standard deviation should be less than 0.4% of the weight range. To maximize the simplification of peripheral circuits and ensure network accuracy, the linearity and symmetry of the memristor resistance modulation curve should be as ideal as possible. The state-of-the-art memristor performance can refer to the previous memristor review.

General memristor metric requirements for edge intelligence are summarized in Table 3. Here, we further discuss the concerns and opportunities in this field.

5.1. Memristor for Edge CNNs

For memristors implemented as basic components of hardware neural networks, more reliable devices are required to improve accuracy, which means higher endurance, retention, yield, and lower variation. Devices with higher resistance states and faster switching speed are preferred to reduce energy consumption and circuit latency, especially in large-scale arrays. Different neural network structures have different levels of tolerance for the nonideal properties of memristors and thus pose different device requirements. Especially, the quantized neural network can greatly alleviate requirements for device performance, which at this stage can already be met by existing devices. For example, the low-bit network has a higher D2D tolerance of the memristor than the full-precision network, which may benefit from regularization. Therefore, the specific structure and algorithm of the network must be fully considered during the device development and optimization. We believe that device
Table 3. Requirements of memristor for edge intelligence.

| Device requirement | Current best (Ideal) | Importance |
|--------------------|----------------------|------------|
| Linearity          | 1.93 (1)[143]        | ★★★☆☆☆ (quantized) |
| Symmetry           | 1.01 (1)[143]        | ★★★☆☆☆ (quantized) |
| D2D variation      | σ = 0.54 (0)[86]     | ★★★☆☆☆ (quantized) |
| C2C variation      | σ/μ = 10% (0)[146]   | ★★★☆☆☆ (quantized) |
| Endurance          | 10^{12} cycles[99]  | ★★★☆☆☆ (in situ) |
| Retention at RT    | > 1000 years[147]    | ★★★☆☆☆ (off situ) |
| Yield              | N.A.                 | ★★★☆☆☆ (quantized) |
| Distinguishable state | 64 levels[92]    | ★★★☆☆☆ (quantized) |
| Stochasticity      | ≈ 9.65%[148]        | ★★★☆☆☆ |
| Switching energy   | 115 fJ[149]          | ★★★☆☆☆ (in situ) |
| Switching speed    | 85 ps                | ★★★☆☆☆ (off situ) |
| Dimensions         | ≈ 2 nm[49]           | ★★★☆☆☆ |

optimization should be conducted on several levels, including a more fundamental understanding of physical mechanisms, the development of CMOS-compatible materials, breakthrough in 3D integration device structures, and development of low-cost fabrication processes.

5.2. Array Design Consideration

It might be a dream of every research group to fabricate a stable and efficient memristive array. Different from previous digital accelerators, the memristive accelerator suffers from insufficient data precision for crossbar inherent defects. The restriction consists of two aspects, array (weight) and peripheral circuit (input/output). For the array limitation, considering device parameters, weight mapping, and low-bit quantization during training process helps recovery accuracy, but the leakage current and IR drop usually cause inaccurate weight or even drift, especially for a high-density, microamp current stage crossbar.[99,152–155] Designing array architectures that inhibit these factors will be an important direction.[99] In addition, ADC and DAC add additional chip overhead and circuit offset for analog manner matrix–vector multiplications.[156] Binary/ternary CNNs with extreme quantization attempts to replace the conversion of analog/digital quantities with simple accumulation, or even simpler logical operations, and compressing the energy-hungry layers first can lower energy consumption.[157,157] Therefore, how to break through the limitations of analog in-memory computing and better integrate with existing mature CMOS technology is worth thinking about.

5.3. Algorithms and Cross-Level Codesign

The industry has positioned INT8 as a more mature quantization scheme.[104] Today, low-bit quantization for edge intelligence shows satisfactory performance, and the combination of quantization and sparsity further reduces weights and operations. But the low-bit quantization of a CNN’s ability to handle complex datasets such as ImageNet shows a loss of information compression that is of concern.[106] The error from each layer might accumulate and cause accuracy loss or nonconvergence.[144] In addition, end-to-end network adaptation for array nonideal factors is an interesting line of thinking. Hybrid training takes more account of the energy and complexity of a mapping device.[30] As for scheme methods such as write validation, efforts have been made in the direction of accurate mapping of weights and values.[131] Unlike the structural optimization of quantized networks, the strategic optimization of the quantization of weight can be achieved without increasing the complexity of the peripheral circuits (the process is implemented in a computer) and improves the network performance, which is a direction worth exploring for quantized networks.[118] The emerging alternating direction method of multipliers (ADMM)-based optimization provides high accuracy in low-bit quantization and structured sparsification.[158–160] The aforementioned point suggests that hardware and software code design algorithms or hardware-specific training methods are more popular in areas such as edge intelligence, where energy efficiency and areas are strictly required.

Acknowledgements

This work was financially supported by the National Key Research and Development Plan of MOST of China (2019YFB2205100, 2016YFA0203800), National Natural Science Foundation of China (No. 61874164, 61841404, 51732003, 61674061), and partially by the Hubei Engineering Research Center on Microelectronics and Chua Memristor Institute.

Conflict of Interest

The authors declare no conflict of interest.

Keywords

convolutional neural network accelerators, edge intelligence, long short-term memory neural networks, memristors, quantization

Received: May 29, 2020
Revised: July 26, 2020
Published online: September 3, 2020

[1] A. Krizhevsky, I. Sutskever, G. E. Hinton, in Advances in Neural Information Processing Systems, Curran Associates, Inc., Red Hook, NY 2012, p. 1097.

[2] T. N. Sainath, A.-r. Mohamed, B. Kingsbury, B. Ramabhadran, in 2013 IEEE Int. Conf. on Acoustics, Speech and Signal Processing, IEEE, Vancouver 2013, p. 8614.
Yi-Fan Qin received his B.S. degree from the School of Optical and Electronic Information, Huazhong University of Science and Technology (HUST) in 2017. He is currently pursuing his M.S. degree. His research focuses on memristive hardware neural networks.

Yi Li is currently an associate professor at Huazhong University of Science and Technology (HUST). He received his Ph.D. degree in microelectronics from HUST in 2014. His research interests include memristors and RRAM and their applications in neuromorphic computing and in-memory computing.
Xiangshui Miao is the founder and director of the Chua Memristor Institute at the Huazhong University of Science and Technology (HUST), where he is currently also a full professor. He received his Ph.D. degree in microelectronics from HUST in 1996. From 1996 to 1997, he was a researcher in City University of Hong Kong. From 1997 to 2007, he was a chief scientist in the Data Storage Institute, A*STAR, Singapore. His research interests include phase change memory, memristors, neuromorphic computing, and in-memory computing.