Review

Resistive-RAM-Based In-Memory Computing for Neural Network: A Review

Weijian Chen [1], Zhi Qi [1], Zahid Akhtar [2] and Kamran Siddique [1,3,*]

[1] School of Computing and Data Science, Xiamen University Malaysia, Sepang 43900, Malaysia
[2] Department of Network and Computer Security, State University of New York Polytechnic Institute, Utica, NY 13502, USA
[3] Department of Computer Science and Engineering, University of Alaska Anchorage, Anchorage, AK 99508, USA
* Correspondence: kamran.siddique@xmu.edu.my

Abstract: Processing-in-memory (PIM) is a promising architecture to design various types of neural network accelerators as it ensures the efficiency of computation together with Resistive Random Access Memory (ReRAM). ReRAM has now become a promising solution to enhance computing efficiency due to its crossbar structure. In this paper, a ReRAM-based PIM neural network accelerator is addressed, and different kinds of methods and designs of various schemes are discussed. Various models and architectures implemented for a neural network accelerator are determined for research trends. Further, the limitations or challenges of ReRAM in a neural network are also addressed in this review.

Keywords: processing-in-memory; in-memory computing; memristor; Resistive Random Access Memory; ReRAM; neural network; CNN; RNN; accelerator

check for updates

Citation: Chen, W.; Qi, Z.; Akhtar, Z.; Siddique, K. Resistive-RAM-Based In-Memory Computing for Neural Network: A Review. *Electronics* **2022**, *11*, 3667. https://doi.org/10.3390/electronics11223667

Academic Editor: Antonio F. Díaz

Received: 1 October 2022
Accepted: 20 October 2022
Published: 9 November 2022

1. Introduction

A major bottleneck in power consumption and performance in the Internet of Things is the data exchange between processor and memory [1]. The commonly used von Neumann architecture separates the processing unit and memory, thus causing a large amount of data movement energy and data access latency [2]. To overcome these limitations, memristor technology emerged, which is a type of Resistive Random Access Memory (ReRAM). Unlike commonly used Random Access Memory, it is able to provide non-volatile storage. ReRAM also offers storage capabilities that are multi-state and have high endurance and high density with much faster access speed [3]. With this, ReRAM is able to overcome the bottleneck faced by conventional CMOS (Complementary Metal Oxide Semiconductor)-based architecture [4]. For example, SRAM (Static Random Access Memory), which is formed by cross-coupled CMOS inverters, has high energy leakage and is unable to efficiently support a wide range of operations in memory [5].

ReRAMs are also relatively small in size, making the integration of memory with computing circuits possible. This leads to an efficient architecture for learning algorithms and neural network applications [6]. With the capabilities of ReRAM, it is a potential element for computing in-memory (CIM), also known as processing-in-memory (PIM). Many proposed ReRAM-based PIM architectures work in the direction of machine learning (ML) accelerators. This is because the ReRAM has the characteristics of an activation-dependent dynamic passive analog device, making it ideal for modeling and updating synaptic weights [6]. Moreover, ReRAM-based processors are able to significantly enhance the efficiency of Convolutional Neural Networks (CNNs) as they involve matrix vector multiplication (MVM), which is supported by ReRAM (Mittal, 2019). This enhancement is achieved because the ReRAM crossbar structure is able to efficiently improve the computational complexity of MVM from $O(n^2)$ to $O(1)$ while eliminating the parameter fetching procedure [7]. Hence, many ReRAM-based PIMs have been proposed to serve the purpose of accelerating neural networks. These include CNNs for image-related tasks, Recurrent Neural Networks (RNNs) and long short-term memory (LSTM) models for audio or voice recognition tasks [8].

The ReRAM device is also known as a metal–insulator–metal (MIM) device, causing it to have an insulator resistance value that can be adjusted [9]. This characteristic of ReRAM causes it to have high potential in machine learning and neural network architectures, as the resistance in ReRAM can be programmed. Resistance modulation is performed on the filament in ReRAM by applying bias to the electrodes, where it can have a different polarity or magnitude, or even both [9]. However, faults can occur in ReRAM, such as an inability to accurately tune the resistance in ReRAM. There are two classifications of ReRAM faults, which are soft faults and hard faults. Soft faults are where the actual tuned resistance is different from the expected value, while hard faults are where the resistance of a ReRAM cell cannot be altered at all [10]. Moreover, neural network training requires large amounts of iterations, where each iteration includes three phases, which are inference, backpropagation and weight updates [11]. With huge training iterations, the weights have to be updated frequently, therefore causing a substantial amount of write operations and meaning that a huge amount of energy is required.

Achieving high throughput and high accuracy are the challenges in ReRAM architecture. It is vital to address these challenges as this will improve the adoption of ReRAM in neural network architectures or neuromorphic computing systems.

2. Background and Related Work

2.1. Processing In-Memory Architecture

The idea of processing in-memory (PIM) was first introduced in the 1960s, where the first computer that used PIM architecture performed processing-in-memory by merging a collection of processing elements and Random Access Memory (RAM). Then, until the early 2000s, it was suggested to join this withthe Dynamic Random Access Memory (DRAM) structure to increase the capability of computation [12]. However, the proposed idea did not gain much recognition in computer architecture as it failed to work practically due to the obstacle of poor technology development during that time. Currently, with the rapid development of technology, research dealing with PIM architectures has been vigorously conducted.

For a better understanding of the PIM architecture, we refer to von Neumann's architecture, in which memory and processing are two separate things. Data are fetched and stored between the CPU and memory, and instead of frequent memory accesses, the cache is introduced in the architecture [13]. The PIM architecture applies the idea of reducing the access rate of memory, and computation is performed in memory rather than in the CPU to reduce the data movement [12]. Due to the limits faced in the DRAM architecture, it became difficult to increase density and reduce energy consumption; therefore, two new approaches for memory system designs have been developed to carry out the PIM architecture, i.e., 3D-stack memory and byte-addressable resistive non-volatile memory (NVM) [12].

Han et al. [14] proposed a ReRAM-based PIM architecture named RPBFS to optimize the graph travelling, specifically on breath first search (BFS), which reduces the data movement on graph traversal. A segment of the memory bank was used to store the graph data, and there was a primary memory bank used to record the data and handle the graph travesal procedure. An algorithm applied on the RPBFS was developed by Han et al. [14] to describe the three phases: graph mapping in one bank, with the information updated in the primary bank; graph initialization, where the status bitmap was initialized; and BFS traversal, where both the graph bank and primary bank worked together.

2.2. Processing-In-Memory vs. Near-Memory Processing

The two approaches mentioned above can be further discussed with the terms in-memory and near-memory. As the name shows, near-memory processing (NMP) refers to the ability to perform computation near the data, thus achieving the aim of speeding up execution. NMP integrates computing cores or PIM logic such as accelerators close to the main memory [12,15]. Three-dimensional stack memory, usually DRAM, contains multiple layers, which include a compute layer for the integration of computing cores [15,16]. Conversely, PIM takes advantage of the original characteristics of the memory cells by interacting and sharing properties between cells themselves to perform the processing [12,15]. Research works have been conducted on the PIM architecture in SRAM, DRAM, phase-change memory (PCM), Magnetic RAM and Resistive RAM, which is the main focus of this paper [12]. Both PIM and NMP are used to reduce unnecessary data movements, but only simple operations such as arithmetic and Boolean logic operations are enabled in the PIM architecture; in addition, even NMP uses the instruction set architecture (ISA) on the computing cores, and the thermal issue of 3D-stack memory has also caused a problem [15].

2.3. Resistive RAM

Metal-oxide resistive RAM (ReRAM or RRAM) is a new memory technology that has been developed recently due to the disadvantages of SRAM and DRAM. Apart from ReRAM, PCM and Magnetic RAM are also being researched as new memory technologies. From [17–19], we have collated Table 1 to show the comparison between existing and modern types of technologies. We can see that ReRAM offers a fast write/read speed compared to other memory devices. Although SRAM has the fastest write/read speed, its cell area is considerably large (at least 25 times that of ReRAM). Compared with ReRAM, the Hard Disk Driver (HDD) has a higher endurance than ReRAM, and its cell area is smaller, but its write/read time is longer [17]. In contrast, ReRAM also has a long duration and low programming voltage while maintaining the cell area [18]. On the other hand, ReRAM has the disadvantage of high energy consumption compared to DRAM, SRAM, STTRAM and Flash, and its energy consumption per bit is tens or even hundreds of times that of other memory devices [19]. However, overall, among all the emerging memory technology candidates, the performance of ReRAM is competitive.

Table 1. Comparisons between different types of memory device [17–19].

Memory Technology	ReRAM	DRAM	SRAM	PCM	STTRAM	Flash (NAND)	HDD
Energy per bit (pJ)	2.7	~0.05	~0.0005	2–25	0.1–2.5	~ 0.00002	$\sim 1 \times 10^5$
Read time (ns)	5	~10	~1	50	10–35	$\sim 1 \times 10^5$	$> 5 \times 10^6$
Write time (ns)	5	~10	~1	10	10–90	$\sim 1 \times 10^5$	$> 5 \times 10^6$
Retention time	$> 3 \times 10^8 s$	~64 ms	-	$> 3 \times 10^8 s$	$> 3 \times 10^8 s$	$> 3 \times 10^8 s$	$> 3 \times 10^8 s$
Voltage (V)	1	<1	<1	1.5	<2	<10	<10
Endurance (circle)	10^{11}	$> 10^{16}$	$> 10^{16}$	$> 10^9$	$> 10^{15}$	$> 10^4$	$> 10^{15}$

There are two main purposes of designing these memory technologies: (I) combining the advantages of SRAM and DRAM, which are the switching speed and storage density, respectively, and (II) the property of non-volatile flash memory [20]. The fundamental structure of the ReRAM cell is generally a metal–insulator–metal form, which can be understood as a sandwich layer with the insulator in between two metal layers [20]. It has various types of chemical compositions built on the three layers, and each layer can be formed with different electrodes for the top and bottom metal layers and oxide materials for the insulator layer [20].

As with other traditional electronic devices, the chosen type of electrode material has a huge influence on the performance of ReRAM. For example, ReRAM with a Cu electrode has a more stable resistive switching behavior than that with a Pt electrode [20]. Generally, the electrode materials utilized by ReRAM can be divided into five different categories based on the composition, which are elementary substance electrodes, silicon-based electrodes, alloy electrodes, oxide electrodes and nitrite-based electrodes [21]. The most frequently used materials are elementary substance electrodes, including Al, CU, Ag, Pt, etc. [22–24].

The state of the ReRAM cell switches with the rise of the conductive filament (CF) inside a dielectric. The CF is a nanoscale component connecting the top and bottom electrodes of the memory cell of ReRAM [25]. According to the type of CF material, ReRAM can be grouped into two kinds: (i) metal ion filament-based ReRAM is referred to as conductive bridge random access memory (CBRAM), and (ii) ReRAM based on an oxygen vacancies filament is known as OxReRAM [25].

Due to the fundamental structure of ReRAM, there is a close relationship between ReRAM and the memristor, which was first introduced to ReRAM by HP in 2008 [26] and attracted a great deal of attention from electronics designers due to its nanometer size and special electrical properties [27]. With the special properties of memristors, the burgeoning memristor-based ReRAM has competitive advantages of high density, low-power, good-scalability, and non-volatility [28]. The concept of the memristor first came from memristor theory, which was put forward earlier in the 1970s. It describes the behaviors of components whose resistances change with the amount of charge flowing through [29], has dramatically influenced the design of ReRAM by providing further theoretical instructions of their behaviors and has been widely used by ReRAM scientists to develop devices and explain corresponding properties more conveniently [30,31].

Payam et al. [27] proposed a type of ReRAM based on the hybrid structure of the memristor and Complementary Metal-Oxide Semiconductor (CMOS). By storing data in the binary or non-binary logic, this design increases the quantities of stored data by increasing the volume of saved data per square area of a memory chip. In their experiment results, the proposed ReRAM design showed a comparable read time and a competitive power consumption with other RAMs [27].

The structure of ReRAM cells can be combined into a crossbar array without much difficulty and distributed into 3D architecture [32]. Due to the crossbar array, ReRAM can perform computation and functions of memory [14]. Though ReRAM has many advantages such as low energy consumption, high density and low operation voltage, it has its challenges as well, such as stability of performance and storage systems on devices with various types of materials [32]. Chi et al. [33] proposed a PIM-based NN accelerator. The accelerator utilized the ReRAM crossbar arrays that can be used to implement Matrix-Vector Multiplication (MVM) in fully connected layers by preprogramming the weight matrix in ReRAM cells to achieve acceleration. The vector is represented by input signals on the worldlines (WLs), each element of the matrix is programmed into the cell conductance in crossbar arrays, and the result of the MVM can refer to the accumulation of the current flowing to the end of each bit line (BL). Because the synaptic weights have already been pre-programmed in ReRAM cells and do not require data to be fetched from the main memory during computation, this highly improves computing efficiency.

Qiu et al. [34] proposed a novel ReRAM crossbar-based (RCA) CNN accelerator, ResiRCA, due to the difficulty of sensing-match latencies in energy-harvesting IoT nodes and the great enhancement of NN accelerator performance in ReRAM. ResiRCA enables the accelerator to fit the energy-harvesting scenario and performs multiplication-and-accumulation (MAC) computations. With the support of the design of lightweight hardware and low power, ResiRCA is allowed to perform scalable computations. Besides, Qiu et al. [34] proposed the ResiSchedule approach, which merges the functions of sequential and pipelined computation modes in order to realize the flexibility of computation scheduling. The approach uses three techniques to achieve high throughput: loop tiling-based computation, ReRAM duplication and inter-layer pipelining.

2.4. Neural Network

The Neural Network (NN) or so-called Artificial Neural Network (ANN) was developed based on the human brain's neural system. It is highly related to Artificial Intelligence (AI). AI is known for its anthropomorphic behavior and ability to learn. Studies on the biological neural system have provided a modeling algorithm for the NN, with the ability to undertake non-linear and parallel tasks such as learning, pattern recognition, memorizing, sensation and control [35].

As mentioned above, the NN architecture uses the idea of the human brain; hence, it is a directed acyclic graph that contains nodes and edges [36]. Figure 1 shows the layers of the NN architecture. L_n refers to the n^{th} layer, while the I_s are the inputs and Q_s are the outputs or results. The layers between the input and output layers are the layers used to perform computations, which are considered as hidden layers or so-called black boxes [37].

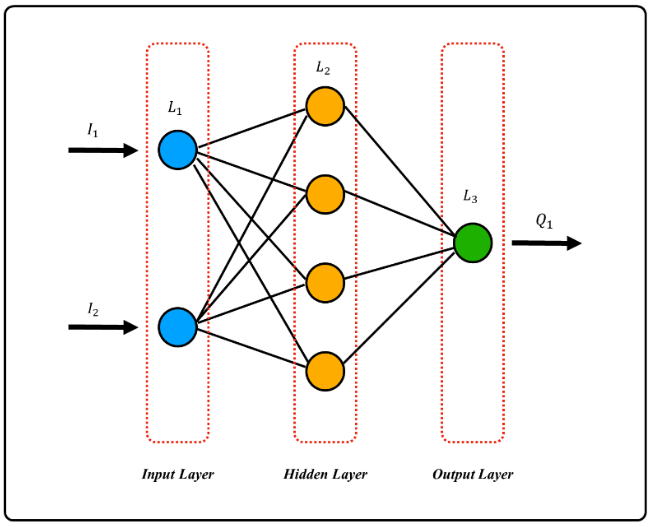


Figure 1. Illustration of basic NN architecture.

There are many existing research works on NN accelerators based on ReRAM that have utilized the memory wall itself and no extra components, greatly reducing the complication of hardware design [38]. Although NN accelerators work well in ReRAM-based architectures with significant efficiency, NN accelerators were unable to fully utilize the advantages of ReRAM due to the inability to satisfy the huge requirements for PE communication, resulting in a communication bottleneck. Therefore, Ji et al. [39] proposed a ReRAM-based NN accelerator based on FPGA, a reconfigurable architecture, to solve the bottleneck problem. Ji et al. [39] found that the weight mapping was imbalanced, where 0.028% of PEs performed 12.5% of computations; therefore, the weight layers were adjusted by duplicating them.

2.5. ReRAM Based Processing-In-Memory Architecture vs. Memristor-Based Neural Network Circuit of Associative Memory

The mainstream opinion [40–42] is that it is better to combine the work on ReRAM and the memristor into one field, since plenty of designs of ReRAM are actual instantiations of the memristor [43]. Hence, apart from the concept of ReRAM-based PIM architecture, other technology based on the memristor, such as the Neural Network circuit design of associative memory, have also attracted researchers' attention. In contrast to the idea of ReRAM-based PIM architecture for reducing the access rate of memory and accomplishing all principal computational tasks in memory [12], the memristor-based Neural Network circuit of associative memory has mainly tended to store the intermediate state data for Artificial Neural Networks [44]. It provides auxiliary functions to simulate human brain neurons to construct more complex Neural Network computing circuits [45,46].

Sun et al. [47] proposed a novel design of the memristor-based Neural Network circuit of Pavlov associative memory, which is a composite system based on a memristor that provides significant biological synaptic functions. This technique simulates learning functions in Artificial Neural Networks by mimicking more realistic situations of memory. In [48,49], the authors further implemented the simulation of the forgetting process among neurons by realizing multiple generalization and differentiation in the memristor-based Neural Network circuit of associative memory.

3. Research Methodology

This paper aims to perform a review of ReRAM-based in-memory processing or computing architecture for neural networks. Challenges and recent research trends regarding Neural Networks using ReRAMs are presented, and various articles and journals relating to the topic are thoroughly studied. The materials were carefully selected based on some criteria listed below.

3.1. Search Strategies

To ensure the review did not lose focus, potential articles and journals were selected based on related keywords. The search string used was as follows:

("ReRAM" *OR* "Resistive RAM" *OR*| "RRAM") *AND* ("neural network architecture" *OR* "NN" *OR* "neural network") *AND* ("in-memory computing" *OR* "processing-in-memory" *OR* "in-memory processing") *AND* "accelerator".

IEEEXplore and ACM (Association for Computing Machinery) Digital Library were the main sources of our selections, as they are able to provide large amounts of quality literature and research papers. However, some articles outside of these sources were also considered with given reliable and quality content.

3.2. Inclusion Criteria

Any articles or research papers with topics regarding ReRAM and Neural Networks were included, specifically those on ReRAM-based accelerators for Neural Networks and also their limitations. Studies on improving Neural Network accelerators using ReRAM or overcoming ReRAM's challenges were emphasized when reviewing various articles.

3.3. Exclusion Criteria

Articles and journals unrelated to Neural Networks or ReRAM were excluded. This was done to focus on the topic and prevent side tracking. In addition, any outdated articles and journals were not included.

3.4. Study Design

The study design of this research is a qualitative systematic literature review. This method was chosen as the purpose of this research is to review the direction of recent studies on NN, specifically on ReRAM-based PIM architecture. This method was conducted in a few steps. Firstly, in the planning stage, research purposes were identified, and a review protocol was drafted and developed. In the conducting stage, research papers for review were selected according to the inclusion and exclusion criteria. Data were extracted by following the guidance in section F (data extraction), and then the quality of the studies was assessed as stated in section E (assessment of quality). Lastly, in the execution stage, data and information were analyzed and distributed.

3.5. Assessment of Quality

The quality of the journals or research papers was evaluated by two reviewers. Each reviewer had to check the titles, abstracts and publication dates to meet the inclusion criteria. To ensure the latest information in the research, journals and research articles earlier than 2015 were rejected.

3.6. Data Extraction

As mentioned, with the aim of reviewing trends and challenges, the following information was extracted:

1. Publication details: The names of the authors and publication year;
2. Model: An overview of the novel model proposed in existing research;

3. Method: The detailed steps and frameworks used for the models to setup experiments;
4. Experiment results: Statistical output of each model for comparison and discussion on the effects; and
5. Future work and challenges: The difficulties faced and flaws detected for future improvements.

3.7. Outcome

The outcome of this review paper is to determine the recent research trends of Neural Networks based on ReRAM-based PIM architecture and the existing limitations of ReRAM-based PIM architecture.

4. Recurrent Neural Network

4.1. ReRAM-Based System Architecture

Long, Na and Mukhopadhyay [2] proposed an RNN accelerator design using ReRAM-based PIM architecture that is different from previous studies where the accelerators were designed for CNN. The purpose of the design of the RNN accelerator was the different features between the CNN and RNN. In the CNN, the inputs are computed independently, and the outputs are based only on its current inputs. However, the outputs are based not only on current inputs but also on previous outputs for RNN. In a previous study [50], the accelerator was only supported for the basic RNN; then, they further proposed an accelerator that supports long short-term memory (LSTM) and the gated recurrent unit (GRU).

The architecture was designed in a similar way to the previous PIM architecture. The memory bank was divided into three sections, which are the memory, buffer and processing subarrays. To make the PIM architecture suitable for the RNN, the processing subarrays were then further divided into three sections again: (1) a ReRAM crossbar subarray for matrix-vector multiplication (MVM), (2) special function unit (SFU) subarrays for non-linear functions, and (3) multiplier subarrays for element-wise operations. In contrast to other works, Long, Na and Mukhopadhyay [2] focused on circuit design: a two-stage Op-amp was used to reduce power consumption, and 16-cycle DAC was used to compute matrix-vector multiplication since the 16-bit fixed point number was divided into 16 individual bits.

The experiment was conducted to evaluate the performance of two RNN applications: Natural Language Processing (NLP) and Human Activity Recognition (HAR). The performance was evaluated on RNN, LSTM, and GRU bases, and the whole experiment was performed on an NVIDIA GPU, with the baseline given. The evaluation of computing was compared with the GPU baseline. As a result, the computing efficiency of this system improved by an average of 79 times, and there was a slight reduction in the accuracy when the standard deviation was less than 0.2.

There are a few challenges when using ReRAM with the RNN accelerator. It is not practical to compute matrix-vector multiplication (MVM) and the element-wise operations with the ReRAM crossbar, as it is the most expensive operation. Furthermore, unnecessary time was spent to wait for the previous output to be calculated from the hidden layers due to the time dependency [2].

4.2. PSB-RNN

Challapalle et al. [51] proposed a ReRAM-based PIM architecture for an RNN accelerator using the block circulant compression approach. The approach achieved the benefits of reducing the computational complexity and space complexity by decomposing the operations into Fourier transforms and point-wise operations. However, the accuracy was reduced slightly. The decomposed operations then were formulated into in-situ Multiply and Accumulations (MAC) operations to provide high throughput. Apart from the block circulant compression, systolic dataflow was employed for communication, improving the energy efficiency and avoiding global data flow. Due to the complex operations, the weight matrices of the block circulant were not able to be mapped directly onto the ReRAM crossbars. Therefore, the mapping was conducted based on the in-situ MAC operations.

The main operation used in this study was matrix-vector multiplication (MVM). The input vectors and previous hidden state vectors in MVM were multiplied with the weight matrices of the block circulant to obtain the output vectors. As mentioned before, MVM was decomposed into discrete Fourier transform and point-wise operations by the fast Fourier transform (FFT) algorithm. The input vector and hidden state vectors were required to be converted into the frequency domain, where its values were pre-calculated and mapped to the crossbar arrays. The procedure for MVM operations was first to compute the input and previous hidden state vectors using FFT and then perform the block-wise element multiplication. Next, the partial products were accumulated, and lastly, the inverse FFT was performed to obtain the MVM result.

Challapalle et al. [51] conducted their experiment by evaluating the computational efficiency. Tensorflow was employed for the implementation of baseline LSTM and GRU networks. In this experiment, CPU and GRU power consumption and the performance evaluation of PSB-RNN were compared with other accelerators, specifically C-LSTM, a Field-Programmable Gate Array (FPGA)-based block circulant and a custom ReRAM crossbar as the baseline architectures. The performance was evaluated on NLP, HAR and TIMIT. PSB-RNN achieved a 44-fold improvement on average in computational efficiency compared to the FPGA-based method and a 17-fold improvement compared to the ReRAM crossbar-based method.

5. Convolutional Neural Network

5.1. ReRAM Memory Wall Accelerator

Chi et al. [33] proposed a PIM-based NN accelerator by simply utilizing the ReRAM crossbar arrays, namely PRIME. Instead of adding an extra processing unit for computation, PRIME takes part of the memory arrays to serve as the accelerator. It fully utilizes the advantages of ReRAM and the PIM architecture in terms of computation capability and accelerates the processing of data. There are other types of memory technologies that can be used for such research, e.g., 3D-stack memory. However, PRIME uses ReRAM instead of 3D-stack memory due to the ease of manufacturing as well as uncomplicated logic of ReRAM. Moreover, the high cost and thermal issues of 3D-stack memory came into consideration. Some other works made use of non-volatile memory technologies to build ternary content addressable memory (TCAM), in which it is necessary to redesign the memory cells to a larger size to support the search operation, which is costly. Therefore, for PRIME, it was decided not to retain the design but to be able to support more advanced computations.

PRIME divided ReRAM into three sections, i.e., memory arrays that are only capable of storage, full function (FF) arrays that contain both computation and storage capabilities and buffer arrays that work with FF arrays as a data buffer. There was a total of 66 subarrays for 2 FF subarrays and 1 buffer subarray per bank by taking 8 levels and 2 levels of input voltage for computation and memory, respectively. By following the input and synapse composting schemes, for computation, 6 bits of dynamic fixed points were given to the input and output, and 8 bits were used for weights. The experiment was conducted by comparing PRIME with pNPU-co and pNPU-pim, using the parallel NPU as a secondary processor and NPU with 3D-stack memory instead of ReRAM.

In addition, four aspects were compared, i.e., speedup performance, execution time breakdown, energy saving and energy breakdown. PRIME had the highest speedup, achieving a 4.1-fold increase compared to pNPU-pim-x86. In contrast, pNPU-co had the lowest result. In the evaluation of execution time breakdown, Chi et al. [33] separated the results into memory and computation. pNPU-pim reduced most of the memory access time from pNPU-co previously, whereas PRIME even reduced it to 0, thus reducing the processing time. Due to the reductions in accessing the memory, PRIME saved a great deal of energy. On the other hand, the energy consumption breakdown of pNPU-co reached almost 100%; however, PRIME's consumption was estimated to be not more than 2%. Generally, PRIME improved the overall performance ~2360-fold and showed an a ~895-fold reduction in energy.

5.2. CNN Accelerator Reusing Data at Multiple Levels

Convolutional Neural Networks (CNN) have contributed greatly towards artificial intelligence applications such as image processing and image classification or image recognition. In addition, ReRAM has great potential for the development of in-memory processing techniques, e.g., CNN accelerators. However, writing to ReRAM takes substantial amount of time compared to reading from ReRAM. Therefore, it is better to map weights from CNN filters to ReRAM and keep it constant until there are any further updates to the weights [52]. This has also caused dataflow control methods and weight mapping schemes to become significant factors in improving processing efficiency.

Luo et al. [52] came up with a ReRAM CNN accelerator called FullReuse. This method aims to achieve the maximum utilization of reusability of weights, input and output data. CNN computing operations are nearly all Multiply and Accumulations (MACs). Out of the total computation in convolutional layers, 90% of them are MACs [53]. This results in massive movements of data due to the large number of weight filters, which also causes large energy consumption. Hence, it is more efficient if data such as input data can be reused instead of reading data from off-chip memory and sending it to ReRAM arrays.

There are a few ways to reuse input data such as convolutional reuse, filter reuse and Input Feature Map (IFM) reuse. Convolutional reuse is where each IFM pixel, during convolution with a filter, is reused except for pixels on the edges. As for filter reuse, when the convolution of multiple IFMs takes place on a filter, the filter weights are able to be reused. While IFM reuse indicates the reuse of each IFM pixel when the particular IFM is convoluted with various different filters, FullReuse uses all data reuse types mentioned, also implementing horizontal and vertical convolutional reuse. To maximize the reuse of data, 3×3 processing elements contain the mappings of filter weights. This shows that this system can simultaneously execute three convolution operations.

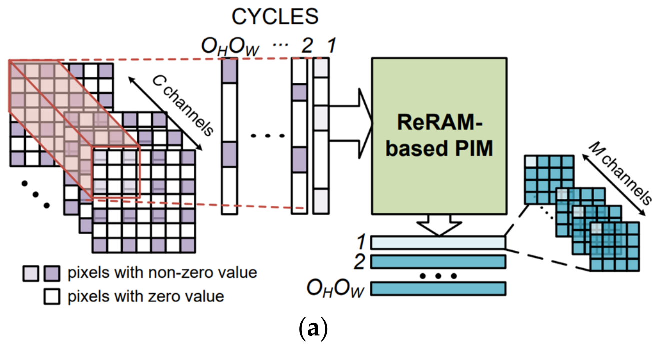
The evaluation of FullReuse is done on a VGG-8 network using a 32 nm DNN+NeuroSim framework. To determine the performance and energy efficiency of the proposed method, CIFAR-10 images were used and processed for FullReuse. Then, the approach is compared to a baseline architecture that adopts a filter mapping scheme where data in the PE on the same row can be reused [54]. The results show that in the layer with the most data movements, FullReuse is able to reduce energy consumption in the buffer and interconnect by 75% and 38%, respectively. As for processing speedup, FullReuse is 1.6 times faster than the baseline architecture.

6. Generative Adversarial Network

6.1. ReRAM-Based Accelerator for Deconvolutional Computation

Z. Li et al. [55] proposed an improved accelerator, known as RED, to improve the execution and energy of deconvolution. Deconvolution is also known as transpose convolution, which is a new type typically used NNs, generative adversarial networks (GANs) and fully convolution networks (FCNs). Deconvolution involves two algorithms that are not seen in traditional convolution: (1) zero padding, which performs padding by adding extra zeros first and then performs convolution; and (2) padding-free, which contains three steps: multiplication, summation and cropping. However, there were some limitations found by Z. Li et al. [55] that led to ineffective computation. With that, Z. Li et al. [55] proposed a novel accelerator using the ReRAM crossbar for computation and optimization.

Figure 2 shows the inefficiency that appeared in deconvolution algorithms. Zero padding inserted extra zeros for the input, causing unnecessary and redundant operations. The padding-free approach based on CMOS can achieve high performance, while in the ReRAM-based PIM, extra storage and significant efforts were needed for circuitry adjustment. Moreover, overlap matrices were generated due to the multiplication of the input and kernel, causing extra calculations and circuits to be needed to obtain the final output. Therefore, the pixel-wise mapping scheme and zero-skipping data flow were suggested and also taken into action by Z. Li et al. [55]. The function of the pixel-wise mapping scheme is to remove unnecessary zeros and provide the ability to execute the computations in parallel. It was designed to match the location of the pixels and support parallel processing. Furthermore, zero-skipping data flow was further expanded to skip the positioning of zeros and maximize the data reuse throughout the processing by distributing the original input to the processing engine (PE) buffers.



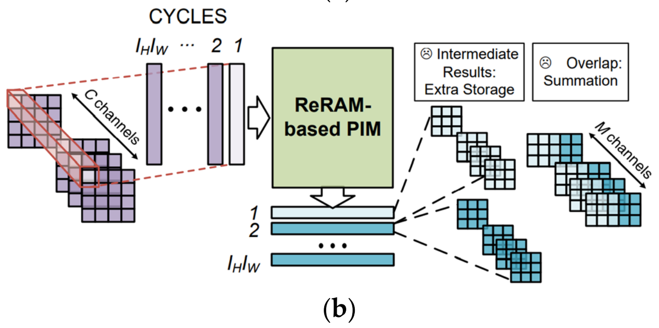


Figure 2. Limitations of the deconvolution algorithm. (**a**) Zero padding and (**b**) padding-free [55].

The experiment was performed by evaluating the execution performance and energy consumption on GAN and FCN models under the zero-padding and padding-free design scenarios. Z. Li et al. [55] executed the RED model by deconvolutional layers; a single layer required multiple processing cycles to complete, and multiple processing modes were executed simultaneously in one cycle. The executing stages were performed as follows: (1) fetch inputs from global memory to the input buffer, (2) process data that passed to Pes, (3) store intermediate output data in the output buffer and (4) transfer back the data to global memory, hence completing the processing cycle. The evaluation was performed by using the modified NeuroSim+, adjusting the clock frequency to 2 GHz and using 2 V of voltage.

RED achieved a 4-fold to 56.16-fold speedup compared to zero-padding, and voc_FCN8s had the highest speedup among all models tested by Z. Li et al. [55]. However, RED only showed slight improvements compared to the padding-free approach. In terms of the execution time breakdown, the data show that RED had the same results for the array and peripheral latency as the padding-free design; the only difference was that the padding-free approach had different output buffer latencies, which can cost an extra 29% to 104% in execution time, but RED did not have this issue in output buffer latency. In terms of energy consumption, RED saved energy 1.05-fold to 18.17-fold compared to the zero-padding design, achieving an 18% to 49% increase in energy efficiency.

6.2. LerGAN

Mao et al. [56] proposed a novel GAN accelerator based on the PIM architecture known as LerGAN. The purpose of LerGAN is to solve two main issues that GAN encountered, i.e., zero-insertion that loads on storage and the complicated data transfer pattern causing the system bottleneck. The problems found in previous studies about GAN accelerators were the unnecessary zero-related works and poor I/O connection. Therefore, LerGAN involved two novel schemes to solve the problems. Mao et al. [56] proposed the zero free data reshaping (ZFDR) scheme and reconfigurable 3D connection scheme to tackle the above problems accordingly. The ZFDR scheme was supported by the memory controller and responsible for solving the zero-related task, whereas the 3D interconnection supports the complicated dataflow.

The design of the ZFDR scheme involves T-CONV ZFDR and W-CONV-S ZFDR components for the purpose of reshaping and reusing weight matrices. The types of each component can be further divided into CornerReshape, where no reuse of reshaped weights occurs; EdgeReshape, where more reuses occur; and InsideReshape, where less reuse occurs. The imbalance of reuse causes an instability of runtime; therefore, EdgeReshape and InsideReshape were replicated to a suitable amount. In the 3D data wire connection unit (3DCU), the wires were used to connect the nodes between different layers. Furthermore, switches were added somewhere between connections due to the bandwidth limitation, and adders were attached to each node, which were Smode and Cmode in 3DCU for memory read/write and computing, respectively.

Considering the two design architectures as a whole, the mapping of reshaped data from ZFDR was basically conducted by the compiler, while 3DCU was used as the memory controller. The two interfaces in ZFDR did not directly reshape the data, but dataflows and place holders were created to remove the zeros. There were mapping generators with both T-CONV and S-CONV and a mapping discriminator with only T-CONV. The data mappings were then recorded by the memory controller for management, and the switches were arranged to update the finite states. The experiment for evaluating the performance was conducted by comparing LerGAN with the FPGA-based GAN, a GAN running on a GPU and a GAN running on an improved ReRAM-based NN accelerator. The hardware used for LerGAN was 4 bits of a ReRAM cell, 16 bits of input, weight and output and a ReRAM array of size 128,128. The benchmark used was a state-of-the-art GAN.

ZFDR improved the overall parallelism by inserting more space for the weights. The transfer inputs was achieved by duplicating the weights after only using up 75% more space. In contrast, 3DUC achieved a reduction of data movements by aligning each part of the phase vertically. Besides, there were two possible results from the two methods: (1) split weights resulted in more spaces saved but less parallelism, and (2) duplicate weights caused more parallelism but used more spaces. The results of the experiment showed the improvements and achievements of LerGAN, where it achieved 47.2-fold, 21.42-fold and 7.46-fold speedup performance compared with the three other architectures, respectively. However, in terms of energy-saving performance, LerGAN consumed 1.04 times more energy than the FPGA-based GAN, but it still resulted in 9.75-fold and 7.68-fold energy savings compared to the other two architectures.

7. Sparse Neural Network

7.1. ReRAM-Based Accelerator for Sparse Neural Network

Lin et al. [7] proposed a mapping and pruning scheme for sparse NNs. Current NNs often bring immense amounts of pressure for storage and computation, as NNs usually contain large numbers of parameters. Hence, sparsity through weight pruning can be used as a powerful approach for compressing the NN model. The method of pruning is able to reduce the redundant parameters, bringing some relief from the storage and computation pressure. Pruning weights can be done by setting individual parameters to zero, which causes the NN to become sparse [57]. However, a sparse matrix still has to be stored in the same way as a dense matrix as the crossbar structure has to retain the proper order of columns and rows. This causes the efforts of weights pruning to be wasted, as the zeros in NN cannot be eliminated. Therefore, Lin et al. [7] came up with a k-means clustering scheme for sparse NN mapping, as shown in Figure 3 which has been adapted from [7], and a crossbar-grained pruning algorithm.

In order to retain the computational complexity of $O(1)$ for matrix-vector multiplication in ReRAM crossbar computing, the whole matrix has to be mapped into the crossbar, even though it includes a large number of zeroes. With k-means clustering, Lin et al. [7] is able to make non-zero elements more concentrated by using column exchanging. However, it is difficult to cluster a very large matrix. Therefore, the matrix goes through a pre-split before performing k-means clustering. The proposed system flow resembles PRIME's system flow, only with extra indexing units added due to the matrix having disordered columns and rows.

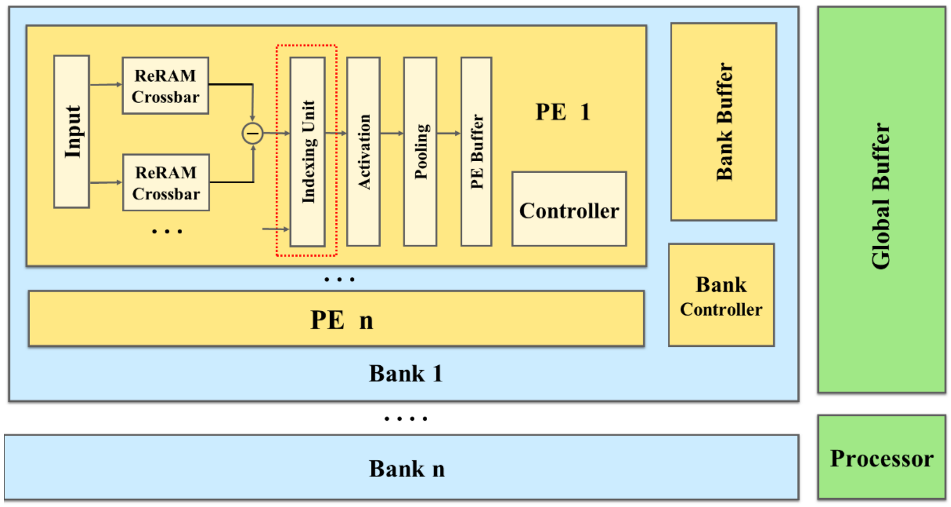


Figure 3. Skeleton of sparse NN mapping scheme system flow chart [7].

After concentrating the non-zero elements as closely as possible, the sparse NN is further compressed using crossbar-grained NN pruning. This allows ReRAM crossbars with only a few parameters to be discarded to minimize hardware resources. Then, the NN is fine-tuned to minimize accuracy loss after weights pruning. Using 8-bit precision, the degrading accuracy with deeper pruning represents less than 1% of accuracy loss.

Different NN structures were included in the stimulation, including CNNs such as ResNet-18 and RNNs such as LSTM-5 (long short-term memory). Based on the results, there was a 3 to 5-fold energy efficiency improvement after crossbar-grained pruning. The results are better for large layered NNs, as the layers occupy a large amount of the energy cost. This results in the NN ending up quite sparse after pruning. The system proposed by Lin et al. [7] also achieved more than a 2.5 to 6-fold speedup for most NNs. However, the result for convolutional layers, e.g., ResNet, is not as good. This is due to it being difficult to compress with pruning and map sparse irregular weights.

7.2. Sparse ReRAM Engine

Several proposed sparse NN/accelerators methods use very ideal architectures of ReRAM such as the PRIME design. The scheme proposed by Lin et al. [7] also uses idealized ReRAM architectures as it is similar to the PRIME system flow, where these optimal or ideal designs assume that the entire crossbar array can be triggered in one cycle. However, this idealized assumption causes a degradation of accuracy towards the inference by NN [58]. This is due to the fact that simultaneously activating a large number of wordlines in the crossbar array causes the analog-to-digital converter (ADC) to have trouble accurately determining the accumulated values on the bitline. For practical applications of ReRAM Neural Network accelerators, a single cycle only allows a limited number of wordlines and bitlines to be turned on. Hence, in a cycle, the accelerator can only carry out a portion of computations.

Yang et al. [58] proposed a method to exploit sparsity in DNNs called Sparse ReRAM Engine (SRE). The maximum number of dot-product computations in one cycle is defined as one Operation Unit (OU). Instead of assuming that the whole of the crossbar array is activated in a single cycle, only a small section of the crossbar or OU is activated in one cycle. With sections of the crossbar turning on independently, this gives some form of advantage in terms of compressing weights and the activation of NNs. The proposed method uses both row compression and activation compression.

Row compression removes the row vectors containing all zero elements in a single OU. Then, the rest of the rows can be shifted up, as shown in Figure 4, which has been adapted from [58]. Column compression is also similar to row compression except that it only involves column operations. However, column compression alters the output mapping of the bitline. Hence, row compression is more suitable as it does not change the output mapping and does not need any output indexing. As for activation compression, a method of Dynamic OU Formation is used, where computation is ignored when the inputs of the OU are all zeros.

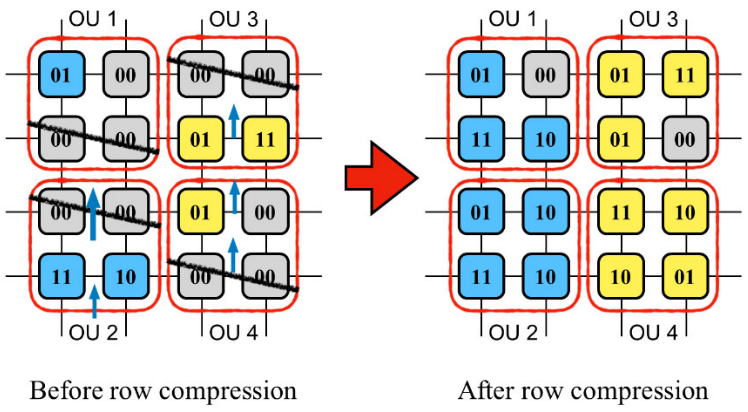


Figure 4. Crossbar array before and after row compression [58].

Comparing SRE with an idealized design (i.e., ISAAC [59]), SRE is able to provide better performance for NN models such as CIFAR-10, CaffeNet and GoogLeNet. In addition, SRE can save an average of 67.0% of energy over the ISAAC design. As for evaluation with other neural network models, SRE is capable of providing an average speedup of 13.1 times and an average energy saving of 85.3%.

8. Deep Neural Network

8.1. HitM: Multi-Modal Deep Neural Network Accelerator

B. Li and Wang et al. [8] proposed an in-memory processing design for multi-modal Deep Neural Network (DNN) acceleration using ReRAM called HitM. Unlike typical unimodal NN applications (e.g., applications dealing with image recognition and natural language processing), the multi-modal DNN is able to learn from heterogeneous data fused from various sensors. For example, videos with auto-captions are some of the data in the web that combine visual and linguistic information [60]. Hence, combining Neural Network architectures such as recurrent and convolutional NNs will result in a multi-modal neural network. This has also caused the utilization of ReRAM-based PIM to be very challenging on multi-modal DNNs. Mainly because they consist of convolutional layers and linear layers inside CNNs and RNNs, respectively, both layers have to be considered to perform optimization on a multi-modal DNN.

HitM mainly consists of two stages, i.e., static analysis and adaptive optimization. The first stage involves static analysis, which mainly extracts structural information of the multi-modal DNN description. It parses the user input neural network description file and generates information of the ReRAM-based PIM design. This information includes the types of NN layers, layer dimension, number of ReRAM crossbars, etc., which is used in the optimization stage. In the adaptive optimization stage, a scheme to perform filter replication is produced with optimized throughput. The rate of production of a network or throughput is evaluated recursively using an estimation model. The next objective is to determine the highest throughput while being under resource constraints, specifically in terms of hardware resources. This can be done by precisely assigning the number of filter replications to each neural layer. Hence, the outputs including the optimized throughput and list of replication numbers can be mapped onto the ReRAM-based PIM.

For the evaluation of hardware and throughput utilization, HitM is compared with ReRAM-based PIM designs without filter replications and an ideal–optimal throughput design. HitM is able to achieve an optimal throughput and hardware utilization of 78.01% and 64.52%, respectively. While comparing with other similar in-memory processing models, HitM is able to improve power efficiency and computation efficiency by 3.80–10.34 times and 4.21–13.08 times respectively. It is expected that the research will extend in the direction of activation in multi-modal NNs and the utilization of weights' bit-widths.

8.2. Re2PIM: Reconfigurable ReRAM-Based Deep Neural Network Accelerator

Deep Neural Network (DNN) computing requires a massive amount of data movement; hence, it demands a large amount of energy and also causes long processing intervals. To overcome the limitations, in-memory processing techniques have emerged to act as accelerators of DNNs. Accelerators using ReRAMs show great capabilities as they are energy efficient and have a high density of computations. However, current ReRAM accelerators are still unable to reach their full potential in terms of computational efficiency. For example, Shafiee et al. [59] determined that the peak computational efficiency for a ReRAM crossbar with a size of 128×128 can go up to 1707 $GOPS/s\times$ mm^2 for 16-bit operations. However, due to peripheral circuits (e.g., buffers, buses and ADCs), an ISAAC architecture can only achieve a computational efficiency of 479 $GOPS/s\times$ mm^2. Other works such as CASCADE and TIMELY architectures are also unable to achieve efficiencies anywhere close to the peak efficiency [61,62]. Hence, overheads in peripheral circuits greatly impact the execution of ReRAM accelerators.

In a DNN accelerator, a ReRAM crossbar, also known as a computing unit (CU), is able to efficiently execute a matrix vector multiplication. However, the size of the CU can affect the energy efficiency and throughput of the accelerators [63]. A small CU size will divide DNN weight matrices into small blocks, resulting in a high overhead of peripheral circuits. For a large CU size, the weight matrices will be divided into larger but fewer blocks, decreasing the overhead. However, a large CU size will limit the throughput of the system due to coarse granular division.

Zhao et al. [63] proposed an accelerator design with a CU size that is variable to adapt to weight matrices with different sizes called Re2PIM. The size of the CU can be changed by reconstructing the reconfigurable units (Rus). Re2PIM is compared to typical ReRAM in-memory architectures, including PRIME, ISAAC and TIMELY. Under constant chip areas, the proposed design of Zhao et al. [63] is able to achieve 27-fold, 34-fold and 1.5-fold average energy efficiency improvements over PRIME, ISAAC and TIMELY, respectively. As for improvements in throughput, Re2PIM is on average 5.7, 17 and 8.2 times better than PRIME, ISAAC and TIMELY, respectively. In terms of crossbar utilization, Re2PIM is slightly worse than ISAAC, as ISAAC has smaller granularity division. However, this causes ISAAC to have high overheads in the peripheral circuit, while Re2PIM benefits from low overhead and saves more energy.

9. Faults in ReRAM Based Neural Networks

Despite the outstanding characteristics of ReRAM over the conventional CMOS architecture, ReRAM still faces some challenges in terms of its reliability. Due to inexperienced chip fabrication and the fundamental nature of ReRAM itself, ReRAM faults will occur [64]. These faults will lead to a degradation of performance and cause errors in the system. ReRAM faults mainly consist of hard faults and soft faults.

Hard faults, which are also known as permanent faults, are where the resistance of ReRAM is stuck or permanently fixed in a state. This is due to a defective wordline or switch state becoming permanent, which means that the ReRAM cell cannot be accessed. This will also lead to the cell being fixed in a permanent state. For example, the cell is mistaken to have high resistance while it is in the low resistance state. This is also known as stuck-at-one [65]. These faulty wordlines or switches can be caused during chip fabrication or due to the low endurance of ReRAM [64].

In contrast, soft faults cause the saved state of ReRAM to be altered, where the actual resistance of the cell differs from the expected resistance, hence causing a degradation of accuracy in the system. ReRAM uses oxygen vacancies and oxygen ions to form conductive filaments. Soft faults are formed when oxygen vacancies are diffused out of the filaments [66]. This can be further triggered by the repeated switching of ReRAM and the generated heat, thus increasing soft faults and causing disturbance in the read and write process.

To alleviate the effects of ReRAM faults, several methods are proposed. For example, Xia et al. [10] proposed a fault-tolerant training that also includes fault detection. The fault-tolerant training will be able to improve the lifetime of ReRAM by reducing the number of write operations and also updating the ReRAM status. When faults are detected through voltage comparison, the next training phase will be able to tolerate the faults.

W. Li et al. [64] presented RRAMedy, which is able to detect ReRAM faults and recover from them. The proposed method uses a detection method in which it can detect subtle faults that may be expensive to detect using traditional methods. This prevents small subtle faults from affecting the accuracy of the system. Moreover, if irreversible faults are detected, the model can be retrained by a cloud server to update the current parameters.

A faulty ReRAM cell can have a major influence on the system if not handled properly. This is because it affects the output map, and this will propagate through the layers of the NN, hence affecting the system as a whole and causing error. The aftereffect of these errors can be disastrous, as it may involve critical safety functions [67]; for example, the classification of objects in self-driving vehicles. Hence, it is important to minimize the effects of ReRAM faults for a system to have higher reliability.

10. Research Trend

Table 2 summarizes the different types of ReRAM-based Neural Network accelerators, including some mentioned in previous sections and other additional recently presented ReRAM-based PIM schemes.

Table 2. Research trend summary.

No	Citation	NN Type	Architecture Summary
1	[2]	RNN	ReRAM-based PIM RNN accelerator whereby the processing engine was divided into three subarrays for different functions, and a lower bit-precision number was utilized to enhance computing efficiency.
2	[7]	Sparse NN	Sparse NN mapping approach using k-means clustering and pruning algorithm to minimize low utilization crossbars with minimal impact on accuracy.
3	[8]	Multimodal-DNN	HitM: Combined NN accelerator model of CNN and RNN by extracting layer-wise information and determining optimized throughput.
4	[33]	MLP and CNN	PRIME: PIM-based architecture using ReRAM crossbar arrays that be configured as the accelerator, with an input and synapse composing scheme.
5	[51]	RNN	PSB-RNN: ReRAM crossbar-based PIM RNN accelerator with systolic dataflow encompassing block circulant compression. This approach has reduced unnecessary space and complexity computation.
6	[52]	CNN	FullReuse: CNN accelerator that aims for maximum utilization of data reusability, including input data, output data and weights.
7	[55]	GAN and FCN	RED: ReRAM-based approach with pixel-wise mapping scheme and zero-skipping data flow for deconvolutional computation. This approach has solved the massive zero insertion and maximizes the reuse of input data.
8	[56]	GAN	LerGAN: PIM-based architecture with zero-free data reshaping scheme and ReRAM-based reconfigurable 3D interconnection.
9	[58]	Sparse NN	Spare ReRAM Engine (SRE) exploiting sparsity in weight and activation in fine-grained operational unit computations.

Table 2. *Cont.*

No	Citation	NN Type	Architecture Summary
10	[61]	CNN/DNN	TIMELY: ReRAM-based PIM accelerators addressing the bottlenecks of the high energy cost of data movement and ADC/DAC processes. This approach adopts analoe local buggers, time-domain interfaces and an only-once input read mapping scheme.
11	[62]	DNN	Re2PIM: DNN accelerator using ReRAM with a reconfigurable size of computing units (CUs) known as reconfigurable units. This design is able to adapt to in-memory computing designs with variably sized matrix vector multiplication.
12	[68]	CNN	PipeLayer: ReRAM-based PIM accelerator that aims to support deep learning applications. Improved pipelined architecture contributed to enabling the continuous flow of data in consecutive cycles. Spike-based data scheme was used to eliminate overhead of DAC/ADC.
13	[69]	GAN	ReGAN: ReRAM-based PIM GAN accelerator with spatial parallelism and computation sharing. This approach reduces memory accesses and increases system throughput with layered computation.
14	[70]	CNN	3DICT: Mobile PIM accelerator that is capable of quality of service (QoS), including high accuracy, low latency and low energy consumption.
15	[71]	GAN	ReRAM-based PIM GAN accelerator with a novel computation deformation technique that completely eliminates zero-insertion. Spatial parallelism and computation sharing has been proposed to improve the training efficiency.
16	[72]	CNN	ReRAM-based network processing unit and training method with mixed precision. This scheme enables low power edge CNN with instant precision tuning.
17	[73]	Spiking NN	A robust ReRAM-based PIM spiking NN accelerator with frequency-dependent stochastic spike-timing-dependent-plasticity (STDP), which achieved better accuracy under noisy input conditions.
18	[74]	DNN	3D-ReG: ReRAM-based PIM architecture and Graphics Processing Unit (GPU) with three-dimensional integration. This design exploits the diversity of 3D-ReG to determine the optimal balance between efficiency and accuracy.
19	[75]	DCNN	Deep Convolutional NN (DCNN) accelerator by avoiding expensive ADC and DAC processes. This is done by moving the computation of network layers to a CMOS peripheral circuit.
20	[76]	RNN	ERA-LSTM: Tiled ReRAM-based PIM with an elaborate mapping scheme that enables the concatenation of tiles for large-scale long short-term memory (LSTM) and inter-tile pipeline. This approach supports LSTM pruning method and saves half of the ReRAM crossbar overhead.
21	[77]	DCN	RECOIN: Deformable Convolution Network (DCN) with low power ReRAM-based PIM architecture by addressing irregular DCN data access. In addition, the authorsproposed a bilinear interpolation (BLI) processing engine to avoid high write latency.

11. Conclusions and Future Studies

Memristor or ReRAM technology has great potential to overcome the limitations faced by current commercial von Neuman architectures using CMOS, including faster access speed and multi-state storage power. The capabilities of ReRAM have also attracted the interest of many researchers in using ReRAM for in-memory computing architectures and building architectures for Neural Networks (NNs). In this paper, we have addressed different approaches and schemes for ReRAM-based NN accelerators. Different types of NNs are categorized into RNN, CNN, GAN, sparse NN and DNN.

The proposed schemes and methods of ReRAM-based NN accelerators mostly aim for higher computation speed, higher energy efficiency and maximum utilization. The accuracy of the ReRAM-based NN is also a significant factor, especially in sparse NN. This is due to sparse NN involving many zero parameters and requiring pruning or compression to reduce storage and computation pressure. However, the challenge comes when the degradation of accuracy must be kept at minimum while increasing efficiency.

Other than accuracy issues caused by compression, ReRAM still poses some risks of resulting in faulty outcomes, as ReRAM has faults such as hard faults and soft faults. These ReRAM faults will affect the resistance stored in the device, leading it to be inaccurate, resulting in bigger errors in the NN. Hence, faults in ReRAM must be addressed properly to minimize their effects on the system.

ReRAM-based NN architectures should be explored more given their powerful capabilities. With this, a large scale of reliable and precise simulators, modeling tools and real prototypes will be needed. However, there is a lack of open-source or non-proprietary modeling tools. This causes many researchers to use existing tools or estimation from other works to determine ReRAM parameters, running the risk of being inaccurate [78]. Hence, the development of non-proprietary tools for ReRAM-based NN testing and prototyping will bring improvements in integrating and adopting ReRAMs in real systems.

Author Contributions: Conceptualization and methodology, W.C., Z.Q. and K.S.; Data curation, W.C and Z.Q.; validation and formal analysis, W.C., Z.Q., Z.A. and K.S.; investigation, W.C., Z.Q., Z.A. and K.S.; writing—original draft preparation, W.C. and Z.Q.; writing—review and editing, W.C., Z.Q., Z.A. and K.S.; Supervision, Z.A. and K.S.; project administration, Z.A. and K.S. and funding acquisition, K.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by Xiamen University Malaysia Research Fund (Grant No: XMUMRF/2022-C9/IECE/0033).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Moreau, M.; Muhr, E.; Bocquet, M.; Aziza, H.; Portal, J.-M.; Giraud, B.; Noel, J.-P. Reliable ReRAM-based Logic Operations for Computing in Memory. In Proceedings of the 2018 IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC), Verona, Italy, 8–10 October 2018; pp. 192–195. [CrossRef]
2. Long, Y.; Na, T.; Mukhopadhyay, S. ReRAM-Based Processing-in-Memory Architecture for Recurrent Neural Network Acceleration. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2018**, *26*, 2781–2794. [CrossRef]
3. Vatwani, T.; Dutt, A.; Bhattacharjee, D.; Chattopadhyay, A. Floating Point Multiplication Mapping on ReRAM Based In-memory Computing Architecture. In Proceedings of the 2018 31st International Conference on VLSI Design and 2018 17th International Conference on Embedded Systems (VLSID), Pune, India, 6–10 January 2018; pp. 439–444. [CrossRef]
4. Halawani, Y.; Mohammad, B.; Abu Lebdeh, M.; Al-Qutayri, M.; Al-Sarawi, S.F. ReRAM-Based In-Memory Computing for Search Engine and Neural Network Applications. *IEEE J. Emerg. Sel. Top. Circuits Syst.* **2019**, *9*, 388–397. [CrossRef]
5. Mittal, S. A survey of architectural techniques for improving cache power efficiency. *Sustain. Comput. Inform. Syst.* **2014**, *4*, 33–43. [CrossRef]
6. Soudry, D.; Di Castro, D.; Gal, A.; Kolodny, A.; Kvatinsky, S. Memristor-Based Multilayer Neural Networks with Online Gradient Descent Training. *IEEE Trans. Neural Networks Learn. Syst.* **2015**, *26*, 2408–2421. [CrossRef]
7. Lin, J.; Zhu, Z.; Wang, Y.; Xie, Y. Learning the sparsity for ReRAM: Mapping and pruning sparse neural network for ReRAM based accelerator. In Proceedings of the 24th Asia and South Pacific Design Automation Conference, New York, NY, USA, 21–24 January 2019; pp. 639–644. [CrossRef]
8. Li, B.; Wang, Y.; Chen, Y. HitM: High-Throughput ReRAM-based PIM for Multi-Modal Neural Networks. In Proceedings of the 2020 IEEE/ACM International Conference on Computer Aided Design (ICCAD), Online, 2–5 November 2020; pp. 1–7.
9. Degraeve, R.; Fantini, A.; Raghavan, N.; Goux, L.; Clima, S.; Govoreanu, B.; Belmonte, A.; Linten, D.; Jurczak, M. Causes and consequences of the stochastic aspect of filamentary RRAM. *Microelectron. Eng.* **2015**, *147*, 171–175. [CrossRef]
10. Xia, L.; Liu, M.; Ning, X.; Chakrabarty, K.; Wang, Y. Fault-Tolerant Training with On-Line Fault Detection for RRAM-Based Neural Computing Systems. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2019**, *38*, 1611–1624. [CrossRef]
11. Cheng, M.; Xia, L.; Zhu, Z.; Cai, Y.; Xie, Y.; Wang, Y.; Yang, H. TIME: A training-in-memory architecture for memristor-based deep neural networks. In Proceedings of the 54th Annual Design Automation Conference 2017, Austin, TX, USA, 18–22 June 2017; pp. 1–6. [CrossRef]
12. Ghose, S.; Boroumand, A.; Kim, J.S.; Gomez-Luna, J.; Mutlu, O. Processing-in-memory: A workload-driven perspective. *IBM J. Res. Dev.* **2019**, *63*, 3:1–3:19. [CrossRef]
13. Connolly, M. *A Programmable Processing-in-Memory Architecture for Memory Intensive Applications*; RIT Scholar Works; Rochester Institute of Technology: Rochester, NY, USA, 2021; Available online: https://scholarworks.rit.edu/theses/10736/ (accessed on 13 October 2021).

14. Han, L.; Shen, Z.; Shao, Z.; Huang, H.H.; Li, T. A novel ReRAM-based processing-in-memory architecture for graph computing. In In Proceedings of the 2017 IEEE 6th Non-Volatile Memory Systems and Applications Symposium (NVMSA), Hsinchu, Taiwan, 16–18 August 2017; pp. 1–6. [CrossRef]
15. Khan, K.; Pasricha, S.; Kim, R.G. A Survey of Resource Management for Processing-In-Memory and Near-Memory Processing Architectures. *J. Low Power Electron. Appl.* **2020**, *10*, 30. [CrossRef]
16. Qi, Z.; Chen, W.; Naqvi, R.A.; Siddique, K. Designing Deep Learning Hardware Accelerator and Efficiency Evaluation. *Comput. Intell. Neurosci.* **2022**, *2022*, 1291103. [CrossRef]
17. Ou, Q.-F.; Xiong, B.-S.; Yu, L.; Wen, J.; Wang, L.; Tong, Y. In-Memory Logic Operations and Neuromorphic Computing in Non-Volatile Random Access Memory. *Materials* **2020**, *13*, 3532. [CrossRef]
18. Varshika, M.L.; Corradi, F.; Das, A. Nonvolatile Memories in Spiking Neural Network Architectures: Current and Emerging Trends. *Electronics* **2022**, *11*, 1610. [CrossRef]
19. Banerjee, W. Challenges and Applications of Emerging Nonvolatile Memory Devices. *Electronics* **2020**, *9*, 1029. [CrossRef]
20. Zahoor, F.; Zulkifli, T.Z.A.; Khanday, F.A. Resistive Random Access Memory (RRAM): An Overview of Materials, Switching Mechanism, Performance, Multilevel Cell (mlc) Storage, Modeling, and Applications. *Nanoscale Res. Lett.* **2020**, *15*, 90. [CrossRef]
21. Gao, S.; Song, C.; Chen, C.; Zeng, F.; Pan, F. Dynamic Processes of Resistive Switching in Metallic Filament-Based Organic Memory Devices. *J. Phys. Chem. C* **2022**, *116*, 17955–17959. [CrossRef]
22. Wu, Y.; Lee, B.; Wong, H.-S.P. Al_2O_3-Based RRAM Using Atomic Layer Deposition (ALD) With 1-μA RESET Current. *IEEE Electron Device Lett.* **2010**, *31*, 1449–1451. [CrossRef]
23. Yang, L.; Kuegeler, C.; Szot, K.; Ruediger, A.; Waser, R. The influence of copper top electrodes on the resistive switching effect in TiO_2 thin films studied by conductive atomic force microscopy. *Appl. Phys. Lett.* **2019**, *95*, 1611–1624. [CrossRef]
24. Chiu, F.-C.; Li, P.-W.; Chang, W.-Y. Reliability characteristics and conduction mechanisms in resistive switching memory devices using ZnO thin films. *Nanoscale Res. Lett.* **2012**, *7*, 178. [CrossRef]
25. Kumar, D.; Aluguri, R.; Chand, U.; Tseng, T.Y. Metal oxide resistive switching memory: Materials, properties and switching mechanisms. *Ceram. Int.* **2017**, *43*, S547–S556. [CrossRef]
26. Strukov, D.B.; Snider, G.S.; Stewart, D.R.; Williams, R.S. The missing memristor found. *Nature* **2008**, *453*, 80–83. [CrossRef]
27. Rabbani, P.; Dehghani, R.; Shahpari, N. A multilevel memristor–CMOS memory cell as a ReRAM. *Microelectron. J.* **2015**, *46*, 1283–1290. [CrossRef]
28. Niu, D.; Xiao, Y.; Xie, Y. Low power memristor-based ReRAM design with Error Correcting Code. In Proceedings of the 17th Asia and South Pacific Design Automation Conference, Sydney, NSW, Australia, 30 January–2 February 2012; pp. 79–84. [CrossRef]
29. Chua, L.O.; Kang, S.M. Memristive devices and systems. *Proc. IEEE* **1976**, *64*, 209–223. [CrossRef]
30. Tedesco, J.L.; Stephey, L.; Hernandez-Mora, M.; Richter, C.A.; Gergel-Hackett, N. Switching mechanisms in flexible solution-processed TiO_2 memristors. *Nanotechnology* **2012**, *23*, 305206. Available online: https://iopscience.iop.org/article/10.1088/0957-4484/23/30/305206/meta (accessed on 13 October 2021). [CrossRef] [PubMed]
31. McDonald, N.R.; Pino, R.E.; Rozwood, P.J.; Wysocki, B.T. Analysis of dynamic linear and non-linear memristor device models for emerging neuromorphic computing hardware design. In Proceedings of the 2010 International Joint Conference on Neural Networks (IJCNN), Barcelona, Spain, 18–23 July 2010; pp. 1–5. [CrossRef]
32. Shen, Z.; Zhao, C.; Qi, Y.; Xu, W.; Liu, Y.; Mitrovic, I.Z.; Yang, L.; Zhao, C. Advances of RRAM Devices: Resistive Switching Mechanisms, Materials and Bionic Synaptic Application. *Nanomaterials* **2020**, *10*, 1437. [CrossRef]
33. Chi, P.; Li, S.; Xu, C.; Zhang, T.; Zhao, J.; Liu, Y.; Wang, Y.; Xie, Y. PRIME: A Novel Processing-in-Memory Architecture for Neural Network Computation in ReRAM-Based Main Memory. In Proceedings of the 2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA), Seoul, Korea, 18–22 June 2016; pp. 27–39. [CrossRef]
34. Qiu, K.; Jao, N.; Zhao, M.; Mishra, C.S.; Gudukbay, G.; Jose, S.; Sampson, J.; Kandemir, M.T.; Narayanan, V. ResiRCA: A Resilient Energy Harvesting ReRAM Crossbar-Based Accelerator for Intelligent Embedded Processors. In Proceedings of the 2020 IEEE International Symposium on High Performance Computer Architecture (HPCA), San Diego, CA, USA, 22–26 February 2020; pp. 315–327. [CrossRef]
35. Babatunde, D.E.; Anozie, A.; Omoleye, J. Artificial Neural Network and Its Applications in The Energy Sector—An Overview. *Int. J. Energy Econ. Policy* **2020**, *10*, 250–264. [CrossRef]
36. Benidis, K.; Rangapuram, S.S.; Flunkert, V.; Wang, B.; Maddix, D.; Turkmen, C.; Gasthaus, J.; Bohlke-Schneider, M.; Salinas, D.; Stella, L.; et al. Neural forecasting: Introduction and literature overview. *arXiv* **2020**, arXiv:abs/2004.10240.
37. Zhang, C.; Liu, Z. Application of big data technology in agricultural Internet of Things. *Int. J. Distrib. Sens. Networks* **2019**, *15*, 1550147719881610. [CrossRef]
38. Nabavinejad, S.M.; Baharloo, M.; Chen, K.-C.; Palesi, M.; Kogel, T.; Ebrahimi, M. An Overview of Efficient Interconnection Networks for Deep Neural Network Accelerators. *IEEE J. Emerg. Sel. Top. Circuits Syst.* **2020**, *10*, 268–282. [CrossRef]
39. Ji, Y.; Zhang, Y.; Xie, X.; Li, S.; Wang, P.; Hu, X.; Zhang, Y.; Xie, Y. FPSA: A Full System Stack Solution for Reconfigurable ReRAM-based NN Accelerator Architecture. In Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems, Association for Computing Machinery, New York, NY, USA, 13–17 April 2019; pp. 733–747. [CrossRef]

40. Gale, E. TiO_2-based Memristors and ReRAM: Materials, Mechanisms and Models. *Semicond. Sci. Technol.* **2014**, *29*, 104004. Available online: https://iopscience.iop.org/article/10.1088/0268-1242/29/10/104004/meta (accessed on 13 October 2021). [CrossRef]
41. Hamdioui, S.; Taouil, M.; Haron, N.Z. Testing Open Defects in Memristor-Based Memories. *IEEE Trans. Comput.* **2015**, *64*, 247–259. [CrossRef]
42. Kim, S.; Jeong, H.Y.; Kim, S.K.; Choi, S.-Y.; Lee, K.J. Flexible Memristive Memory Array on Plastic Substrates. *Nano Lett.* **2011**, *11*, 5438–5442. [CrossRef]
43. Gale, E.; Pearson, D.; Kitson, S.; Adamatzky, A.; Costello, B.d.L. Aluminium electrodes effect the operation of titanium oxide sol-gel memristors. *arXiv* **2011**, arXiv:1106.6293v2.
44. Jo, S.H.; Chang, T.; Ebong, I.; Bhadviya, B.B.; Mazumder, P.; Lu, W. Nanoscale Memristor Device as Synapse in Neuromorphic Systems. *Nano Lett.* **2010**, *10*, 1297–1301. [CrossRef] [PubMed]
45. Pan, C.; Hong, Q.; Wang, X. A Novel Memristive Chaotic Neuron Circuit and Its Application in Chaotic Neural Networks for Associative Memory. *IEEE Trans. Comput. Des. Integr. Circuits Syst.* **2021**, *40*, 521–532. [CrossRef]
46. Sun, J.; Han, G.; Zeng, Z.; Wang, Y. Memristor-Based Neural Network Circuit of Full-Function Pavlov Associative Memory with Time Delay and Variable Learning Rate. *IEEE Trans. Cybern.* **2020**, *50*, 2935–2945. [CrossRef]
47. Sun, J.; Han, J.; Liu, P.; Wang, Y. Memristor-based neural network circuit of pavlov associative memory with dual mode switching. *AEU Int. J. Electron. Commun.* **2021**, *129*, 153552. [CrossRef]
48. Sun, J.; Han, J.; Wang, Y.; Liu, P. Memristor-Based Neural Network Circuit of Emotion Congruent Memory With Mental Fatigue and Emotion Inhibition. *IEEE Trans. Biomed. Circuits Syst.* **2021**, *15*, 606–616. [CrossRef]
49. Sun, J.; Wang, Y.; Liu, P.; Wen, S.; Wang, Y. Memristor-Based Neural Network Circuit with Multimode Generalization and Differentiation on Pavlov Associative Memory. *IEEE Trans. Cybern.* **2022**. [CrossRef]
50. Long, Y.; Jung, E.M.; Kung, J.; Mukhopadhyay, S. ReRAM Crossbar based Recurrent Neural Network for human activity detection. In Proceeding of the 2016 International Joint Conference on Neural Networks (IJCNN), Vancouver, BC, Canada, 24–29 July 2016; pp. 939–946. [CrossRef]
51. Challapalle, N.; Rampalli, S.; Chandran, M.; Kalsi, G.; Subramoney, S.; Sampson, J.; Narayanan, V. PSB-RNN: A Processing-in-Memory Systolic Array Architecture using Block Circulant Matrices for Recurrent Neural Networks. In Proceeding of the 2020 Design, Automation & Test in Europe Conference & Exhibition (DATE), Grenoble, France, 9–13 March 2020; pp. 180–185. [CrossRef]
52. Luo, C.; Diao, J.; Chen, C. FullReuse: A Novel ReRAM-based CNN Accelerator Reusing Data in Multiple Levels. In Proceeding of the 2020 IEEE 5th International Conference on Integrated Circuits and Microsystems (ICICM), Nanjing, China, 23–25 October 2020; pp. 177–183. [CrossRef]
53. Chen, Y.-H.; Krishna, T.; Emer, J.S.; Sze, V. Eyeriss: An Energy-Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks. *IEEE J. Solid-state Circuits* **2017**, *52*, 127–138. [CrossRef]
54. Peng, X.; Liu, R.; Yu, S. Optimizing Weight Mapping and Data Flow for Convolutional Neural Networks on RRAM Based Processing-In-Memory Architecture. In Proceeding of the 2019 IEEE International Symposium on Circuits and Systems (ISCAS), Sapporo, Japan, 26–29 May 2019; pp. 1–5. [CrossRef]
55. Li, Z.; Li, B.; Fan, Z.; Li, H. RED: A ReRAM-Based Efficient Accelerator for Deconvolutional Computation. *IEEE Trans. Comput. Des. Integr. Circuits Syst.* **2020**, *39*, 4736–4747. [CrossRef]
56. Mao, H.; Song, M.; Li, T.; Dai, Y.; Shu, J. LerGAN: A Zero-Free, Low Data Movement and PIM-Based GAN Architecture. In Proceeding of the 2018 51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), Fukuoka, Japan, 20–24 October 2018; pp. 669–681. [CrossRef]
57. Bandaru, R. Pruning Neural Networks. Towards Data Science. 2 September 2020. Available online: https://towardsdatascience.com/pruning-neural-networks-1bb3ab5791f9#:~{}:text=Neural%20network%20pruning%20is%20a,removing%20unnecessary%20neurons%20or%20weights (accessed on 20 November 2021).
58. Yang, T.-H.; Cheng, H.-Y.; Yang, C.-L.; Tseng, I.-C.; Hu, H.-W.; Chang, H.-S.; Li, H.-P. Sparse ReRAM Engine: Joint Exploration of Activation and Weight Sparsity in Compressed Neural Networks. In Proceedings of the 2019 ACM/IEEE 46th Annual International Symposium on Computer Architecture (ISCA), Phoenix, AZ, USA, 22–26 June 2016; pp. 236–249. [CrossRef]
59. Shafiee, A.; Nag, A.; Muralimanohar, N.; Balasubramonian, R.; Strachan, J.P.; Hu, M.; Williams, R.S.; Srikumar, V. ISAAC: A Convolutional Neural Network Accelerator with In-Situ Analog Arithmetic in Crossbars. In Proceedings of the 2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA), Seoul, Korea, 18–22 June 2016; pp. 14–26. [CrossRef]
60. Bernardi, R.; Cakici, R.; Elliott, D.; Erdem, A.; Erdem, E.; Ikizler-Cinbis, N.; Keller, F.; Muscat, A.; Plank, B. Automatic description generation from images: A survey of models, datasets, and evaluation measures. *J. Artif. Intell. Res.* **2016**, *55*, 409–442. Available online: https://dl.acm.or (accessed on 13 October 2021). [CrossRef]
61. Chou, T.; Tang, W.; Botimer, J.; Zhang, Z. CASCADE: Connecting RRAMs to Extend Analog Dataflow in An End-To-End In-Memory Processing Paradigm. In Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture, Columbus, OH, USA, 12–16 October 2019; pp. 114–125. [CrossRef]
62. Li, W.; Xu, P.; Zhao, Y.; Li, H.; Xie, Y.; Lin, Y. TIMELY: Pushing Data Movements and Interfaces in PIM Accelerators towards Local and in Time Domain. In Proceedings of the ACM/IEEE 47th Annual International Symposium on Computer Architecture, Online, 30 May–3 June 2020; pp. 832–845. [CrossRef]

63. Zhao, Y.; He, Z.; Jing, N.; Liang, X.; Jiang, L. Re2PIM: A Reconfigurable ReRAM-Based PIM Design for Variable-Sized Vector-Matrix Multiplication. In Proceedings of the 2021 on Great Lakes Symposium on VLSI, Online, 22–25 June 2021; pp. 15–20. [CrossRef]
64. Li, W.; Wang, Y.; Li, H.; Li, X. RRAMedy: Protecting ReRAM-Based Neural Network from Permanent and Soft Faults During Its Lifetime. In Proceedings of the 2019 IEEE 37th International Conference on Computer Design (ICCD), Abu Dhabi, Unite Arab Emirates, 17–20 November 2019; pp. 91–99. [CrossRef]
65. Chen, C.-Y.; Shih, H.-C.; Wu, C.-W.; Lin, C.-H.; Chiu, P.-F.; Sheu, S.-S.; Chen, F.T. RRAM Defect Modeling and Failure Analysis Based on March Test and a Novel Squeeze-Search Scheme. *IEEE Trans. Comput.* **2015**, *64*, 180–190. [CrossRef]
66. Tosson, A.M.; Yu, S.; Anis, M.H.; Wei, L. Analysis of RRAM Reliability Soft-Errors on the Performance of RRAM-Based Neuromorphic Systems. In Proceedings of the 2017 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), Bochum, Germay, 3–5 July 2017; pp. 62–67. [CrossRef]
67. Li, G.; Hari, S.K.S.; Sullivan, M.; Tsai, T.; Pattabiraman, K.; Emer, J.; Keckler, S.W. Understanding Error Propagation in Deep Learning Neural Network (DNN) Accelerators and Applications. In Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, Denver, CO, USA, 12–17 November 2017; pp. 1–12. [CrossRef]
68. Song, L.; Qian, X.; Li, H.; Chen, Y. PipeLayer: A Pipelined ReRAM-Based Accelerator for Deep Learning. In Proceedings of the 2017 IEEE International Symposium on High Performance Computer Architecture (HPCA), Austin, TX, USA, 4–8 February 2017; pp. 541–552. [CrossRef]
69. Chen, F.; Song, L.; Chen, Y. ReGAN: A pipelined ReRAM-based accelerator for generative adversarial networks. In Proceedings of the 2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC), Jeju Island, Korea, 22–25 January 2018; pp. 178–183. [CrossRef]
70. Lou, Q.; Wen, W.; Jiang, L. 3DICT: A Reliable and QoS Capable Mobile Process-In-Memory Architecture for Lookup-based CNNs in 3D XPoint ReRAMs. In Proceedings of the 2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), San Diego, CA, SUA, 5–8 November 2018; pp. 1–8. [CrossRef]
71. Chen, F.; Song, L.; Li, H. Efficient process-in-Memory architecture design for unsupervised GAN-based deep learning using ReRAM. In Proceedings of the 2019 on Great Lakes Symposium on VLSI, Tysons Corner, VA, USA, 9–11 May 2019; pp. 423–428. [CrossRef]
72. He, Y.; Wang, Y.; Wang, Y.; Li, H.; Li, X. An Agile Precision-Tunable CNN Accelerator based on ReRAM. In Proceedings of the 2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), Westminster, CO, USA, 4–7 November 2019; pp. 1–7. [CrossRef]
73. She, X.; Long, Y.; Mukhopadhyay, S. Improving robustness of reram-based spiking neural network accelerator with stochastic spike-timing-dependent-plasticity. In Proceedings of the 2019 International Joint Conference on Neural Networks (IJCNN), Budapest, Hungary, 14–19 July 2019. [CrossRef]
74. Li, B.; Doppa, J.R.; Pande, P.P.; Chakrabarty, K.; Qiu, J.X.; Li, H. 3D-ReG: A 3D ReRAM-based Heterogeneous Architecture for Training Deep Neural Networks. *ACM J. Emerg. Technol. Comput. Syst.* **2020**, *16*, 1–24. [CrossRef]
75. Zheng, Q.; Wang, Z.; Feng, Z.; Yan, B.; Cai, Y.; Huang, R.; Chen, Y.; Yang, C.-L.; Li, H.H. Lattice: An ADC/DAC-less ReRAM-based Processing-In-Memory Architecture for Accelerating Deep Convolution Neural Networks. In Proceedings of the 2020 57th ACM/IEEE Design Automation Conference (DAC), Online, 20–24 June 2020; pp. 1–6. [CrossRef]
76. Han, J.; Liu, H.; Wang, M.; Li, Z.; Zhang, Y. ERA-LSTM: An Efficient ReRAM-Based Architecture for Long Short-Term Memory. *IEEE Trans. Parallel Distrib. Syst.* **2020**, *31*, 1328–1342. [CrossRef]
77. Chu, C.; Chen, F.; Xu, D.; Wang, Y. RECOIN: A Low-Power Processing-in-ReRAM Architecture for Deformable Convolution. In Proceedings of the 2021 on Great Lakes Symposium on VLSI, Online, 22–25 June 2021; pp. 235–240. [CrossRef]
78. Mittal, S. A Survey of ReRAM-Based Architectures for Processing-In-Memory and Neural Networks. *Mach. Learn. Knowl. Extr.* **2019**, *1*, 5. [CrossRef]