An Energy-Efficient Digital ReRAM-Crossbar-Based CNN With Bitwise Parallelism

Ni, Leibin; Liu, Zichuan; Yu, Hao; Joshi, Rajiv V.

2017

Ni, L., Liu, Z., Yu, H., & Joshi, R. V. (2017). An Energy-Efficient Digital ReRAM-Crossbar-Based CNN With Bitwise Parallelism. IEEE Journal on Exploratory Solid-State Computational Devices and Circuits, 3, 37-46.

https://hdl.handle.net/10356/85536

https://doi.org/10.1109/JXCDC.2017.2697910

An Energy-efficient Digital ReRAM-crossbar based CNN with Bitwise Parallelism

Leibin Ni, *Student Member, IEEE*, Zichuan Liu, Hao Yu, *Senior Member, IEEE*, and Rajiv V. Joshi

***Abstract*—There is great attention to develop hardware accelerator with better energy efficiency as well as throughput than GPUs for convolutional neural network (CNN). The existing solutions have relatively limited parallelism as well as large power consumption (including leakage power). In this paper, we present a ReRAM-accelerated CNN that can achieve significantly higher throughput and energy efficiency when the CNN is trained with binary constraints on both weights and activations, and is further mapped on a digital ReRAM-crossbar. We propose an optimized accelerator architecture tailored for bitwise convolution that features massive parallelism with high energy efficiency. Numerical experiment results show that the binary CNN accelerator on a digital ReRAM-crossbar achieves a peak throughput of 792 GOPS (Giga operations per second) at the power consumption of 4.5 mW, which is 1.61 times faster and 296 times more energy-efficient than a high-end GPU.**

***Index Terms*—Approximate computing, Neural network hardware, Nonvolatile memory, Resistive RAM, Supervised learning**

I. Introduction

Convolutional neural network (CNN) has become a promising machine learning engine for image-oriented data analytics [1]. A GPU-based CNN accelerator is currently dominant in use. It can achieve high throughput in convolution but with high power consumption [1]. On the other hand, a FPGA-based CNN accelerator has also been investigated due to its energy efficiency benefits [2] but it has quite limited low parallelism with the need of reduced numeric precision. Moreover, for image-data oriented computing, a large amount of data needs to be hold in memory with significant leakage power consumption. As such, it really requires a re-examination of both of the CNN algorithm as well the underlying hardware platform towards high energy efficiency as well as high throughput in convolution.

The recent advancement in binary-constrained deep learning [3] has introduced new insight for a more efficient hardware acceleration of the CNN. The work in [3] demonstrates the successful use of binarized weights in CNN with training under the binary constraints. As such, highly parallel bitwise operation can be realized in hardware with great potential to out-performed GPUs. However, there is little work exploring on the hardware accelerator architecture for binary convolutional neural networks. It is noticed that the main computation here involves intensive bitwise operations such as convolution, batch normalization, pooling and activation functions. Moreover, the traditional CPU/GPU-based acceleration is out of memory with arithmetic cores that have poor bandwidth and energy efficiency, not to mention the standby power.

The recent emerging non-volatile memory (NVM) technologies have shown significantly reduced standby power and increased integration density, as well as close-to DRAM/SRAM access speed. In addition, studies in [4], [5], [6] have shown the logic implementation based on NVM devices. The resistive random access memory (ReRAM) devices [7] have shown a great potential for an energy-efficient acceleration, especially for a natural mapping of multiplier on crossbar. It can be employed as both storage and computation element with minimized leakage power due to its non-volatility [8]. The traditional CPU/GPU-based acceleration is out of memory with arithmetic cores that have poor bandwidth to access memory, low efficiency to process data, not to mention the huge standby power of memory. A digital ReRAM-crossbar [9] can further support the binary matrix-vector multiplication even under strong non-uniformity with process variation when compared to the analog ReRAM-crossbar computing with additional ADCs [10], [11], [12]. The digital ReRAM-crossbar implementation is also shown in [13] for single layer feed forward network, which still has limitation on large datasets such as CIFAR-10 [14]. Compared to the work in [15], this paper evaluates the system on a larger-scale benchmark CIFAR-10 instead of MNIST [16], and performs more energy-efficiency analysis. Moreover, previous works [9], [10], [12] only show the realization of the DOT (matrix-vector-multiplication) operation used in convolution with only exploration on MNIST benchmark. There is no study on how to realize a bitwise matrix-vector multiplication, batch normalization, pooling function, and activation function all on the ReRAM-crossbar devices.

In this paper, based on the digital ReRAM-crossbar, we have developed a bitwise convolutional neural network based image-data processing using CIFAR-10 benchmark. We show that the digital ReRAM-crossbar can be used for a bitwise convolution, batch normalization, pooling function and activation function, all in ReRAM-crossbar devices. Moreover, the intermediate results between steps are not required to be written back to the memory with little standby power. The inference accuracy is observed high under the ReRAM device variation as well as the binary constrained results.

This revised manuscript is submitted on April 22, 2017. This work is sponsored by grants from Singapore NRF-CRP (NRF-CRP9-2011-01, NRF-CRP16-2015-03) and MOE Tier-2 (MOE2015-T2-2-013).

L. Ni, Z. Liu, and H. Yu are all with School of Electrical and Electronic Engineering, Nanyang Technological University (NTU), Singapore. E-mail: haoyu@ntu.edu.sg

R. V. Joshi is with IBM T. J. Watson Research Center, Yorktown Heights, New York.

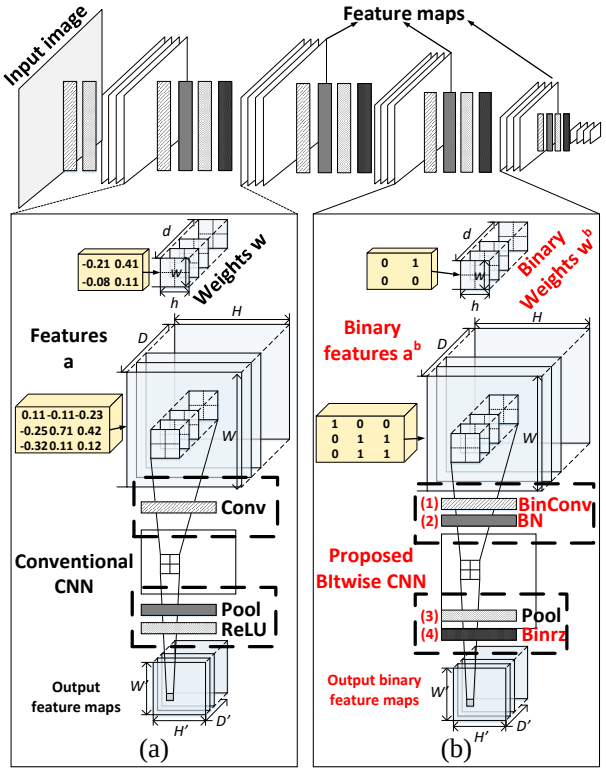


Fig. 1: Computation flow comparison between (a) conventional CNN; and (b) bitwise-parallelized CNN.

The rest of this paper is organized as follows. The bitwise CNN (BCNN) operations are compared with conventional CNN in Section II. The in-memory ReRAM accelerator architecture and digital ReRAM-crossbar background is introduced in Section III. The mapping between BNN and digital ReRAM-crossbar is discussed in Section IV. Numerical experiment results are presented in Section V with conclusion drawn in Section VI.

II. CNN WITH BITWISE PARALLELISM

The recent work in [3] suggests a CNN using binary constraints during training. In this section, we will discuss how to generate a CNN model with bitwise parallelism for convolution, batch normalization, pooling function and activation function.

A. Bitwise Convolution

The convolution is the most time-consuming and computation-intensive operation in CNN. The binary convolution in [3] uses $\{-1,+1\}$ for both input features and weights, so that the floating-point matrix-vector-multiplication operation is not required. However, the negative binary weights cannot be directly realized on hardware. To avoid the negative weights, recent work [17] uses bitwise XNOR and bit-count operation in $\{0,1\}$ for hardware implementation. The bitwise CNN (BCNN) can be represented as follows:

$$s_k(x,y,z) = \sum_{i=1}^{w_k}\sum_{j=1}^{h_k}\sum_{l=1}^{D_k} w_k^b(i,j,l,z) \otimes a_{k-1}^b(i+x-1, j+y-1, l), \quad (1)$$

where $w_k^b \in \{0,1\}^{w_k \times h_k \times D_{k-1} \times D_k}$ is the binary weight in k-th block, $a_{k-1}^b \in \{0,1\}^{W_{k-1} \times H_{k-1} \times D_{k-1}}$ is the binary input feature map and also the output of the binary convolution, and $\otimes$ is defined as bitwise XNOR operation. Comparing to a real-valued CNN in the single-precision data format, since the elements of weights and feature maps can be stored in 1-bit, both of the logic and memory resources required for binary convolutional (BinConv) layer can be greatly reduced. Meanwhile, it can lead to a higher parallelism as well as greater energy-efficiency improvement.

B. Bitwise Batch Normalization

Next, batch normalization is required to stabilize and accelerate the training process. In the inference stage, the normalization is retained to match training process. The output of the normalization can be represented by

$$a_k(x,y,z) = \frac{s_k(x,y,z) - \mu(x,y,z)}{\sqrt{\sigma^2(x,y,z)}}\gamma(x,y,z) + \beta(x,y,z), \quad (2)$$

where $\mu \in \mathbb{R}^{W_k \times H_k \times D_k}$ and $\sigma^2 \in \mathbb{R}^{W_k \times H_k \times D_k}$ are the expectation and variance over the mini-batch, while $\gamma \in \mathbb{R}$ and $\beta \in \mathbb{R}^{W_k \times H_k \times D_k}$ are learnable parameters [18] that scale and shift the normalized value. In the inference stage, μ, σ^2, γ and β are all fixed to normalize the convolution output.

C. Bitwise Pooling and Activation Functions

The pooling layer performs a down-sampling across a $M \times M$ contiguous region on the feature map output by normalization layer. Pooling is used for selecting the most significant information from the features. It also provides translation invariance and reduces the computation intensity. Two kinds of pooling schemes are commonly used in CNN. One is the max-pooling, which takes the maximum value of the pooling region. The other is average-pooling, which takes the mean value of the pooling region. In this paper, the binary max-pooling is applied with the following equation:

$$a_k'(i,j) = Max\{a_k(Mi+k, Mj+l)\}, (k,l \in (0,M)), \quad (3)$$

where a_k and a_k' are the features before and after pooling, respectively. The activation function to process the output of max-pooling is called binarization (Binrz), which can be represented as

$$a_k^{b'}(i,j) = \begin{cases} 0, & a_k'(i,j) \le 0 \\ 1, & otherwise \end{cases} \quad (4)$$

To summarize pooling and binarization, we can observe that these two steps are to find out the sign of the maximum number in the pooling region. As such, we can do the binarization for all the numbers in the pooling region first, and then find the maximum among them. Since the results from binarization become only 0 or 1, the pooling process only needs to detect if there is any 1 in the region. An example with the exchange of pooling and binarization for a 2×2 region is shown below:

$$\begin{bmatrix} +0.70 & -0.42 \\ -0.21 & +0.35 \end{bmatrix} \Longrightarrow \begin{cases} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} & \Longrightarrow 1 \\ 0.70 & \Longrightarrow 1 \end{cases} \quad (5)$$

Here, the 2×2 real-value matrix denotes the output of batch normalization. The first path in (5) is doing binarization first, then max-pooling, while the second path is doing it conversely. It is clear that exchanging the max-pooling and binarization will not affect the final output in the inference stage.

D. Bitwise CNN Model Overview

The overall working flow of a bitwise-parallelized CNN model is shown in Figure 1(b). Each BinConv layer takes the binary feature map generated from the previous layer as input and conducts a bitwise convolution between binary feature maps and binary filter weights. The convolution output is further processed by the normalization layer before down-sampling by a the max-pooling layer. The down-sampled feature maps are subsequently fed into Binrz layer that produces binary non-linear activations according to the input sign.

The key difference between the developed bitwise CNN and the direct-truncated CNN using less precision bit [19] is illustrated as follows. The direct-truncated CNN is obtained by reducing the numerical precision in post-training phase while the bitwise CNN developed here is obtained by training with binary constraints [3]. As such, the direct-truncated CNN that suffers from accuracy loss in general, but the bitwise CNN retains most of the accuracy with lowest precision. In the inference stage, only the binarized weights w_k^b will be retained for much smaller storage, faster inference, higher parallelism as well as higher energy-efficiency. The BNN algorithm is detailed in supplementary file.

III. Digital ReRAM for In-memory Computing

In this section, basics of ReRAM device and digital ReRAM-crossbar are reviewed to support an in-memory computing architecture, which will be used to map the bitwise CNN discussed in Section II.

A. ReRAM Device

The emerging resistive random access memory (ReRAM) [20], [21] is a two-terminal device for memory storage with 2 non-volatile states: high resistance state (HRS) R_{off} and low resistance state (LRS) R_{on}. Besides working as memory storage, a ReRAM-crossbar can be applied to perform logic operations. In one ReRAM-crossbar, given the input probing voltage V_{WL} on each write-line (WL), the current I_{BL} on each bit-line (BL) becomes the natural multiplication-accumulation of current through each ReRAM device. Therefore, the ReRAM-crossbar array can intrinsically perform the analog matrix-vector-multiplication [12] by

$$\begin{bmatrix} I_{BL,1} \\ \vdots \\ I_{BL,M} \end{bmatrix} = \begin{bmatrix} c_{1,1} & \cdots & c_{1,M} \\ \vdots & \ddots & \vdots \\ c_{N,1} & \cdots & c_{N,M} \end{bmatrix} \begin{bmatrix} V_{WL,1} \\ \vdots \\ V_{WL,N} \end{bmatrix} \tag{6}$$

where $c_{i,j}$ is configurable conductance of the ReRAM resistance $R_{i,j}$, which can represent real number of weights. Compared to the traditional CMOS implementation, the ReRAM-crossbar achieves higher level of parallelism and consumes less power, including standby power.

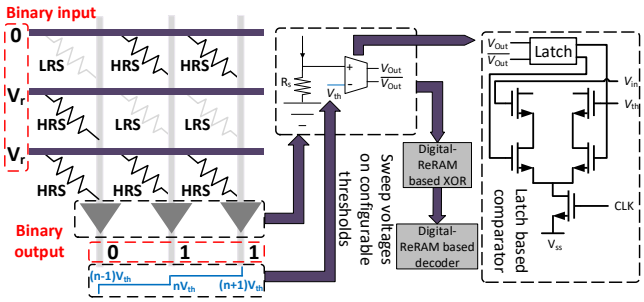


Fig. 2: Digital ReRAM-crossbar with 1-bit voltage comparator configured with ladder-like threshold voltage.

B. Digital ReRAM-crossbar

Previous mapping of CNN are mainly based on the traditional analog ReRAM-crossbar [10], [12]. The main limitation is that there exists a huge non-uniform analog resistance for undetermined states in the analog computation, which can result in convolution error [22]. In addition, a large overhead is required in ADC conversions.

As shown in Figure 2, the recent work in [9] introduces a digital ReRAM-crossbar for roust digital matrix-vector multiplication. In the digital ReRAM-crossbar, only 0 or V_r is applied on the input world-line (WL), and only RHS or LRS is configured for each ReRAM device. To overcome the sneak-path problem, half voltage operating scheme [23] is applied. For the output bit-line (BL), a sense amplifier (SA) is applied to identify whether the output is 0 or 1. Note that the key difference here is the threshold V_{th} of SA in each BL can be configured in a ladder-like voltages. As such, the sensed analog output can be encoded in binary to produce the multiplied result [9]. Such a binary matrix-vector multiplication can still be used in applications such as [24].

Compared to the traditional analog ReRAM-crossbar [10], [12], the digital ReRAM-crossbar has advantages in the following perspectives:

- It has better programming accuracy of the ReRAM device under process variation with no additional ADC conversation as well.
- Since only LRS and HRS are configured in digital ReRAM-crossbar, it does not require high HRS/LRS ratio so that low-power (high LRS) device [21] can be applied.

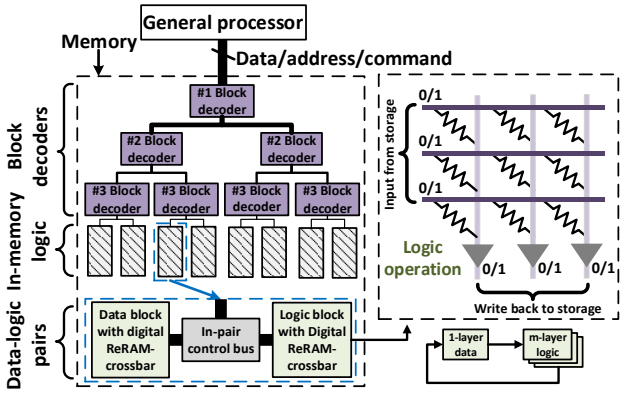


Fig. 3: In-memory computing architecture based on digital ReRAM-crossbar.

In addition, the wire resistance affects little on IR-drop when using the low-power ReRAM device.
- The binary input voltage has a better robustness on the IR-drop in large-size crossbar.

Moreover, there is no work exploring on how to map all the CNN operations such as normalization, pooling and activation functions on the ReRAM devices. This paper will show details in Section IV on how to map all bitwise CNN operations on the digital ReRAM devices.

C. In-memory Computing Architecture

Based on the digital ReRAM-crossbar, one can develop an in-memory computing architecture with both memory and logic implemented by the ReRAM-crossbar as shown in Figure 3. In this architecture, data-logic pairs are located in a distributed fashion, where data transmission among data block, logic block and external scheduler are maintained by a control bus. As such, logic block can read data locally and write back to the same data block after the logic computation is done [9]. As a result, the huge I/O communication load between memory and general processor can be relieved because most of the data transmission is done inside data-logic pairs. Based on this in-memory computing architecture using the digital ReRAM-crossbar, we will introduce the main contribution of this paper in next section: how to map all of the bitwise CNN operations?

IV. Bitwise CNN on Digital ReRAM-crossbar

In this section, we will focus on the mapping of all the bitwise CNN operations on the digital ReRAM-crossbar such as convolution, batch normalization, pooling and activation functions.

A. Mapping Bitwise Convolution

According to (1), the bitwise convolution can be split into several XNOR and bit-count results of two vectors. To implement (1), we can use two AND operations for a^b_{k-1} and w^b_k as well as their complements. Therefore, the bitwise convolution on ReRAM-crossbar can be shown as follows:

$$s_k = \sum_{i=1}^{N} w^b_k \otimes a^b_{k-1} = \sum_{i=1}^{N} (w^b_k \cdot a^b_{k-1} + \overline{w^b_k} \cdot \overline{a^b_{k-1}}) \quad (7)$$

The mapping of the bitwise convolution is shown in Figure 4(a). It requires a $2N \times N$ ReRAM-crossbar, where N is the number of element in the vector. All columns are configured with the same elements that correspond to one column in binary weight w^b_k of the neural network, and the WL voltages are determined by the binary input a^b_{k-1}. Due to the large ratio between R_{off} and R_{on}, the current through the BL is approximately equal to $\frac{s_k V_r R_s}{R_{on}}$, where s_k is the inner-product result in (1). Since the current of all BLs is identical, the ladder-like threshold voltages $V_{th,j}$ are set as follows:

$$V_{th,j} = \frac{(2j+1)V_r R_s}{2R_{on}}, \quad (8)$$

TABLE I: Binary format stored in ReRAM-crossbar

BinConv output	Batch normalization	Floating binary format
0	-0.5	10111111000000000000000000000000
1	-0.3	10111110100110011001100110011010
...	...	.....
7	0.9	00111111011001100110011001100110

where $V_{th,j}$ is the threshold voltage for the j_{th} column. If we use s' to denote the output array, and $s'(j)$ to denote the output of column j in ReRAM-crossbar, we can have

$$s'_k(j) = \begin{cases} 1, & j < s_k \\ 0, & j \geq s_k \end{cases} \quad (9)$$

In this case, the inner-product results s_k can be recognized that the first $(N - s_k)$ output bits are 0 and the rest s_k bits are 1. The relation between s'_k and s_k can be expressed as $s'_k = g(s_k)$.

As described in (1), each binary weight vector w_k performs bitwise convolution with several input features. As a result, each logic block in Figure 3 stores a binary vector w_k, while the control bus transmits the input feature sequentially. In this case, bitwise convolution can be performed in parallel in separated logic blocks.

B. Mapping Bitwise Batch Normalization

Bitwise batch normalization requires two digital ReRAM-crossbars in the implementation. In the first ReRAM-crossbar, it performs the XOR operation on adjacent bits of the output of bitwise convolution. It can be expressed as

$$s''_k(j) = \begin{cases} 1, & j = s_k \\ 0, & j \neq s_k \end{cases} \quad (10)$$

After that, the second ReRAM-crossbar builds a look-up-table (LUT). Since μ, σ^2, γ and β are all fixed in the inference stage, (2) can be rewritten as

$$a_k = f(s_k), \quad (11)$$

where $f(\cdot)$ represents the LUT. As a result, the LUT is stored in the second ReRAM-crossbar according to the parameters μ, σ^2, γ and β. As described in (10), only the s_k-th row of the LUT is selected, so the batch normalization result can be directly readout. The threshold voltage of both two ReRAM-crossbars are

$$V_{th} = \frac{V_r R_s}{2R_{on}} \quad (12)$$

To have a better illustration, Figure 4(b) shows the detailed mapping and Table I shows the values to store in the second ReRAM-crossbar when $\mu = 2.5$, $\sigma^2 = 5$, $\gamma = 1$ and $\beta = 0$ referred to the IEEE-754 standard.

C. Mapping Bitwise Pooling and Binarization

According to (5), we will do the binarization first and then perform max-pooling. The bitwise activation in (4) can be achieved by selecting the sign-bit of the binary format output of Figure 4(b). In max-pooling, the output is 0 only when

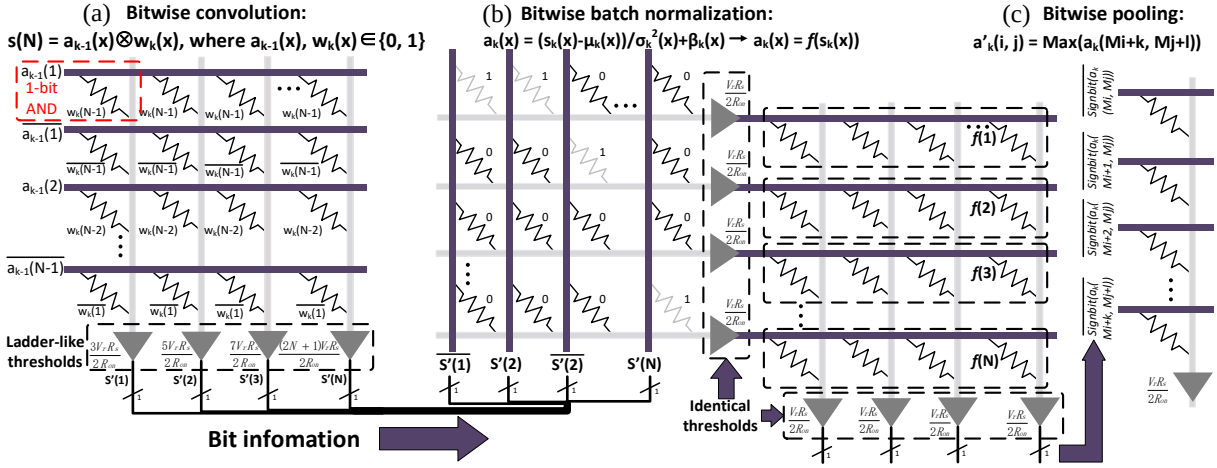


Fig. 4: Digital ReRAM-crossbar mapping for (a) bitwise convolution, (b) bitwise batch normalization, (c) bitwise max-pooling.

all the numbers in the pooling region are negative. As a result, we can add all the complementary of sign-bit in the pooling region. If the result is not 0, it indicates that at least one positive number is in the pooling region, resulting in the pooling result 1. In summary, the max-pooling in (3) can be re-written as

$$a'_k(i,j) = \begin{cases} 1, \text{if} \sum \overline{Sign\{a_k(Mi+k, Mj+l)\}} > 0 \\ 0, \text{otherwise} \end{cases} \quad (13)$$

As a result, both of the bitwise max-pooling and binarization can be implemented by the addition operation performed on the digital ReRAM-crossbar with 1 column, as shown in Figure 4(c).

D. Summary of Mapping Bitwise CNN

As a summary, the four operations of the bitwise CNN in Figure 1(b) can be fully mapped onto the digital ReRAM-crossbar. All the threshold voltages are fixed even if the parameters of bitwise CNN are changed. Although these operations are implemented in different ReRAM-crossbar array, the input/output formats of them are compatible so that one can directly connect them as shown in the logic block in Figure 3 with pipeline used. The pipeline design is based on Table II, so that each stage implements a layer. CONV-2, CONV-4 and CONV-6 are the stages which require the most ReRAM cells and computation time. As a result, more logic blocks are assigned to these steps to relieve the critical path. In our simulation, half of the digital ReRAM crossbars are used to perform these three layers. Moreover, since the output feature from Figure 4(c) is binary, the area overhead and energy consumption of the data storage can be significantly reduced. In addition, because layers in Table II are implemented in different data-logic pairs, the volume of data transmitted is also decreased.

V. Numerical Result

A. Simulation Settings

1) Baselines: In the simulation, we have implemented different baselines for comparison using both MNIST and CIFAR-10 benchmarks. The detail of each baseline is listed below:

CPU: The BNN simulation is run in Matconvnet [25] on a computer server with 3.46GHz core and 64.0GB RAM. The BNN network is referred to Table II.

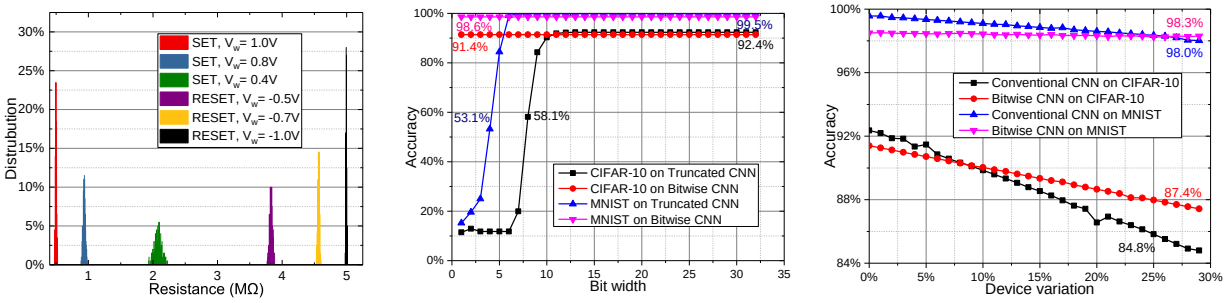
Our design of Digital-ReRAM: The ReRAM device model for BNN is based on [21], [26], [27] with the resistance of ReRAM set as 0.5M and 5M as on-state and off-state respectively with working frequency of 200MHz. Sense amplifier is based on the design of [28]. The BNN network is also referred to Table II.

Others: GPU-based [3] Bitwise CNN implementations and FPGA-based [2], CMOS-ASIC based [29] and Analog-ReRAM based [12] conventional CNN implementations are selected for performance comparisons as well.

2) Network: The overall network architecture of the bitwise CNN (called BNN) is shown in Table II. It has six binary convolutional layers (BinConv), three max-pooling layers (MP) and three fully-connected layers (FC). It takes a 32×32 RGB image as the input of the first layer. We use 128 sets of binary filters, and each set contains 3 binary filters to process the data from R, G, B channels, receptively. The width and height of each BinConv layer is fixed to 3×3 with stride of 1 and zero padding of 1. The BinConv layer performs the bitwise convolution between input feature maps and weights followed by the bitwise batch normalization and the binary activation.

TABLE II: Bitwise CNN configuration of CIFAR-10 dataset

Name	Filter/weight	Output Size
CONV-1	$3 \times 3 \times 3 \times 128$	$128 \times 32 \times 32$
CONV-2	$3 \times 3 \times 128 \times 128$	$128 \times 32 \times 32$
MP-2	2×2	$128 \times 16 \times 16$
CONV-3	$3 \times 3 \times 128 \times 256$	$256 \times 16 \times 16$
CONV-4	$3 \times 3 \times 256 \times 256$	$256 \times 16 \times 16$
MP-4	2×2	$256 \times 8 \times 8$
CONV-5	$3 \times 3 \times 256 \times 512$	$512 \times 8 \times 8$
CONV-6	$3 \times 3 \times 512 \times 512$	$512 \times 8 \times 8$
MP-6	2×2	$512 \times 4 \times 4$
FC-1	8192×1024	1024
FC-2	1024×1024	1024
FC-3	1024×10	10



(a) (b) (c)

Fig. 5: (a) ReRAM configuration with voltage amplitude variation following Gaussian distribution (b) accuracy comparison with analog ReRAM under device variation (c) accuracy comparison with truncated CNN with approximation.

Note that the computation for FC layers can be treated as a convolution with stride of 0 and zero padding of 0. Thus, we refer the matrix multiplication in FC layers to convolution as well. As shown in Table II, two cascaded convolutional layers form a convolutional block with equivalent 5×5 convolution window. This configuration provides more powerful representation capacity with less amount of weights compared with the direct 5×5 implementation. Meanwhile, binary batch-normalization is applied after the convolution to accelerate and stabilize the training. We adopt the binary activation to each BinConv layer and FC layer with the exception for the last FC layer. The output of the last FC layer is fed into the softmax layer [30] without binarization to generate a probabilistic distribution of 10 classes.

B. Accuracy Comparison

We first show accuracy comparison between the analog-ReRAM and the digital-ReRAM under device variation. We then show accuracy comparison between the conventional CNN with direct-truncated prevision and the proposed bitwise CNN. Various benchmarks such as CIFAR-10 [14] and MNIST [16] are used here.

1) Error under Device Variation: In the previous discussion, the digital ReRAM-crossbar has better programming accuracy than the analog ReRAM-crossbar. Figure 5(a) shows a 200-time Monte-Carlo simulation of a single ReRAM device programing process with different write voltage V_w, where the voltage amplitude is under Gaussian distribution on the($3\sigma = 3\%V_w$), and each column denotes a region of $5k\Omega$. It is clear that the digital ReRAM-crossbar (only $500k\Omega$ and $5M\Omega$) can achieve a better uniformity than the analog ReRAM-crossbar. The accuracy comparison against device variation on ReRAM is shown in Figure 5(b). Monte-Carlo is applied in the generation of device variation. In CIFAR-10, one can observe that when the device variation (ReRAM resistance value) is more than 8%, there is a large output current error reported. For example, when the device variation reaches 29%, the digital ReRAM can have an accuracy of 87.4% with only 4% decreased compared to no variations, better than the analog one with an accuracy of 84.8% with 7.6% decreased. In MNIST, the digital ReRAM is always better even when the device variation is larger than 27%.

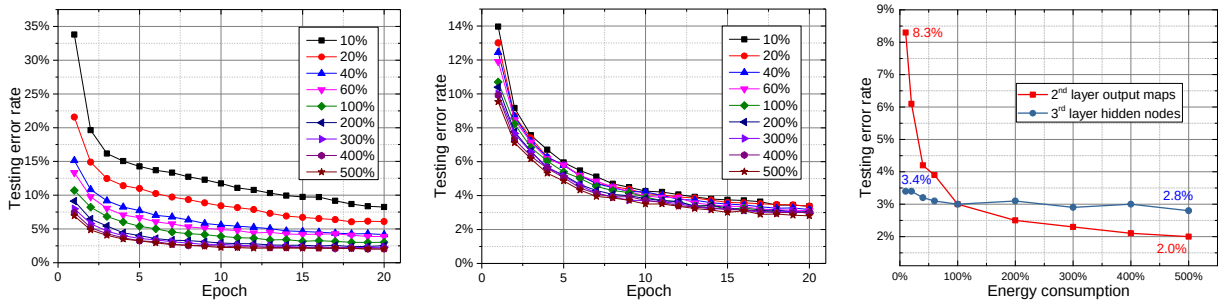
TABLE III: Bitwise CNN configuration of MNIST dataset

Name	Filter/weight	Output Size
CONV-1	$5 \times 5 \times 1 \times 20$	$20 \times 28 \times 28$
MP-1	2×2	$20 \times 14 \times 14$
CONV-2	$5 \times 5 \times 20 \times 50$	$50 \times 14 \times 14$
MP-2	2×2	$50 \times 7 \times 7$
FC-1	2450×500	500
FC-2	500×10	10

2) Error under Approximation: For a precision-bit direct-truncated CNN, we use the conventional approach to train the full-precision (32-bit) CNN first, and then decrease the precision of all the weights in the network. The numerical experiment results of weights with different bit width is shown in Figure 5(c). Here the weights in the bitwise CNN is 1 bit, whose accuracy of is not changed. In CIFAR-10, although the accuracy of the full precision (32-bit) can reach 92.4% in the direct-truncated CNN, the bit-width influences the accuracy a lot especially when the bit-width is smaller than 10. For example, when the precision decreases to 6-bit, the accuracy drops down to only about 11.8%. In MNIST, the accuracy of the direct-truncated CNN drops significantly when the bit-width is smaller than 6-bit. The results show that the proposed bitwise CNN can perform much better than the direct-truncated CNN.

C. Scalability Study

To achieve a better energy-efficiency of BNN, we do the scalability study to find out the BNN parameters for both good testing accuracy and energy-efficiency. We use a 4-layer BNN on MNIST benchmark as a baseline (100% energy-efficiency, as shown in Table III), and change the number of output maps of layer 2 (CONV-2, 50) and hidden nodes of layer 3 (FC-1, 500), as shown in Figure 6(a) and 6(b). For each energy-efficiency configuration, we do a 20-epoch training to make a fair comparison. When the number of hidden nodes or output maps decreases, the energy-efficiency is better but it will cause higher testing error rate. To summarize the scalability study, Figure 6(c) shows that the hidden nodes of layer 2 is more sensitive to testing accuracy. As a result, increasing the hidden nodes of layer 2 is better for higher accuracy while decreasing the hidden nodes of layer 3 is better for energy-efficiency.



(a) (b) (c)

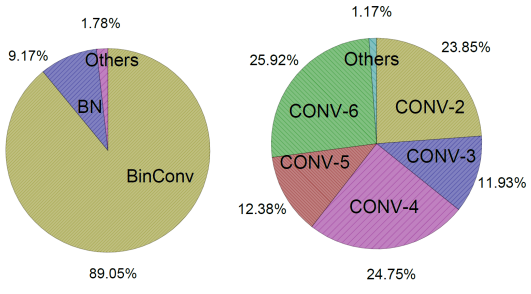
Fig. 6: Scalability study of testing error rate under different energy consumption with (a) layer 2 output maps, and (b) layer 3 hidden nodes; (c) comparison with layer 2 output maps and layer 3 hidden nodes.

D. Performance Comparison

In this section, $1,000$ images with 32×32 resolution in CIFAR-10 are selected to evaluate the performance among all implementations. Parameters including binary weights, and LUT for batch normalization have been configured by the training process. The detailed comparison is shown in Table IV with numerical results including area, system throughput, computation time and energy consumption. In the numerical experiment, every batch comprises of 10 images and the results are calculated in parallel. The detailed result figure is included in supplementary file. The black square represents to the class that each image belongs.

1) Power: The overall power comparison among all the implementations are shown in Table IV under the similar accuracy of 91.4% in CIFAR-10. Compared to CPU-based and GPU-based bitwise CNN, the proposed digital-ReRAM based implementation can achieve up to four-magnitude smaller power. Moreover, compared to FPGA-based and CMOS-ASIC based conventional CNN, the digital-ReRAM based implementation is $4,155$ times and 62 times smaller power.

In addition, we analyze the detailed power characteristics of the proposed digital-ReRAM design in Figure 7. Firstly, we analyze the power distribution on convolution, batch normalization, pooling and activation, respectively. Figure 7(a) shows that 89.05% of power is consumed by convolution, while 9.17% of power is for batch normalization, and the rest only takes 1.78%. Secondly, we analyse the power distribution on different bitwise CNN layers in Table II. Results in Figure 7(b) show that CONV-6 consumes the most power 25.92%, while CONV-4 and CONV-2 also consume more than 23% of the total power. In general, there is over 98% of power consumed by CONV-2 to CONV-6 layers.



(a) (b)

Fig. 7: Power consumption of bitwise CNN (a) for different operations (b) for different layers.

2) Throughput and Efficiency: For the throughput performance, we use GOPS (Giga operations per second) to evaluate all the implementations. The proposed digital-ReRAM can achieve 792 GOPS, which is 535 times and 1.61 times better than CPU-based and GPU-based implementations, respectively. It is also 12.78 times and 18.86 times better than FPGA-based and CMOS-ASIC. For energy-efficiency, the digital-ReRAM achieves 176 TOPS/W, three-magnitude better than the CMOS-ASIC based CNN. In area-efficiency comparison, the digital-ReRAM is 296 times better than CMOS-ASIC. The digital-ReRAM is the best among all the implementations on both throughput and efficiency.

The design exploration of CIFAR-10 is shown in Figure 8. We change the number of CONV-4 output maps to find out the optimized parameter. The normalized energy consumption is referred to the configuration in Table IV. The result shows that the accuracy will not increase when the number is larger than 256. When it is lower than 256, the accuracy drops down even though it has better energy consumption and throughput. As a result, the parameters in Table II are optimal with specifications in Table IV.

VI. Conclusion

In this paper, we propose a digital ReRAM-crossbar based in-memory accelerator for bitwise convolutional neural network. The bitwise CNN is obtained by training with binary constraints so that operations including convolution, batch

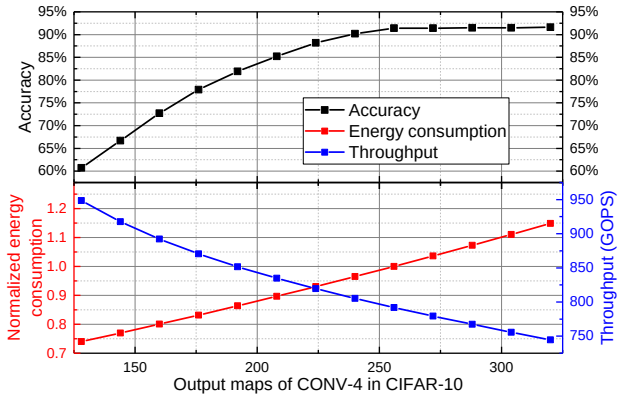


Fig. 8: Performance with different number of CONV-4 output maps in CIFAR-10.

TABLE IV: Performance comparison under different software and hardware implementations

Implementation	CPU (Matconvnet)	GPU [3]	FPGA [2]	CMOS-ASIC [29]	Analog ReRAM [12]	Digital ReRAM
Network category	Bitwise CNN	Bitwise CNN	CNN	CNN	CNN	Bitwise CNN
Frequency	3.46GHz	1.05GHz	100MHz	250MHz	-	100MHz
Area	240mm^2	601mm^2	-	12.25mm^2	-	0.78mm^2
Average power	130W	170W	18.7W	278mW	-	4.5mW
System throughput (GOPS)	1.48	493	62	42	-	792
Frame per second (FPS)	1.2	400	46	35	-	643
Energy-efficiency	0.011GOP/J	2.9GOP/J	3.32GOP/J	151GOP/J	2TOP/J	176TOP/J
Area-efficiency	0.006GOP/s/mm^2	0.82GOP/s/mm^2	-	3.43GOP/s/mm^2	-	1015GOP/s/mm^2

- refers to the data not reported.

normalization, pooling and activation can be implemented with bitwise parallelism. The bitwise CNN can be further effectively mapped to the digital ReRAM-crossbar with significant speed-up and energy-efficiency improvement. Numerical results using the benchmark CIFAR-10 show that the developed bitwise CNN accelerator on the digital ReRAM-crossbar achieves a peak throughput of 792 GOPS at the power consumption of 4.5 mW, which is 1.61 times faster and 296 times more energy-efficient than existing state-of-arts.

REFERENCES

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

[2] C. Zhang, P. Li, G. Sun, Y. Guan, B. Xiao, and J. Cong, "Optimizing fpga-based accelerator design for deep convolutional neural networks," in *Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*. ACM, 2015, pp. 161–170.

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

[4] Y. Wang, L. Ni, C.-H. Chang, and H. Yu, "Dw-aes: A domain-wall nanowire-based aes for high throughput and energy-efficient data encryption in non-volatile memory," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 11, pp. 2426–2440, 2016.

[5] A. Sharma, T. Jackson, M. Schulaker, C. Kuo, C. Augustine, J. Bain, H.-S. Wong, S. Mitra, L. Pileggi, and J. Weldon, "High performance, integrated 1t1r oxide-based oscillator: Stack engineering for low-power operation in neural network applications," in *VLSI Technology (VLSI Technology), 2015 Symposium on*. IEEE, 2015, pp. T186–T187.

[6] S.-C. Chang, N. Kani, S. Manipatruni, D. E. Nikonov, I. A. Young, and A. Naeemi, "Scaling limits on all-spin logic," *IEEE Transactions on Magnetics*, vol. 52, no. 7, pp. 1–4, 2016.

[7] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, "The missing memristor found," *nature*, vol. 453, no. 7191, pp. 80–83, 2008.

[8] Y. Wang, H. Yu, L. Ni, G.-B. Huang, M. Yan, C. Weng, W. Yang, and J. Zhao, "An energy-efficient nonvolatile in-memory computing architecture for extreme learning machine by domain-wall nanowire devices," *Nanotechnology, IEEE Transactions on*, vol. PP, no. 99, pp. 1–1, 2015.

[9] L. Ni, H. Huang, Z. Liu, R. V. Joshi, and H. Yu, "Distributed in-memory computing on binary rram crossbar," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 13, no. 3, p. 36, 2017.

[10] M. Hu, J. P. Strachan, Z. Li, E. M. Grafals, N. Davila, C. Graves, S. Lam, N. Ge, J. J. Yang, and R. S. Williams, "Dot-product engine for neuromorphic computing: programming 1t1m crossbar to accelerate matrix-vector multiplication," in *Proceedings of the 53rd Annual Design Automation Conference*. ACM, 2016, p. 19.

[11] Y. Wang, H. Yu, and W. Zhang, "Nonvolatile cbram-crossbar-based 3-d-integrated hybrid memory for data retention," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 5, pp. 957–970, 2014.

[12] L. Xia, T. Tang, W. Huangfu, M. Cheng, X. Yin, B. Li, Y. Wang, and H. Yang, "Switched by input: power efficient structure for rram-based convolutional neural network," in *Proceedings of the 53rd Annual Design Automation Conference*. ACM, 2016, p. 125.

[13] L. Ni, H. Huang, and H. Yu, "On-line machine learning accelerator on digital rram-crossbar," in *Circuits and Systems (ISCAS), 2016 IEEE International Symposium on*. IEEE, 2016, pp. 113–116.

[14] A. Krizhevsky, V. Nair, and G. Hinton, "The cifar-10 dataset," 2014.

[15] S. Yu, Z. Li, P.-Y. Chen, H. Wu, B. Gao, D. Wang, W. Wu, and H. Qian, "Binary neural network with 16 mb rram macro chip for classification and online training," in *Electron Devices Meeting (IEDM), 2016 IEEE International*. IEEE, 2016, pp. 16–2.

[16] Y. LeCun, C. Cortes, and C. J. Burges, "Mnist handwritten digit database," *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, 2010.

[17] Z. Liu, Y. Li, F. Ren, and H. Yu, "A binary convolutional encoder-decoder network for real-time natural scene text processing," *arXiv preprint arXiv:1612.03630*, 2016.

[18] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[19] J. Qiu, J. Wang, S. Yao, K. Guo, B. Li, E. Zhou, J. Yu, T. Tang, N. Xu, S. Song *et al.*, "Going deeper with embedded fpga platform for convolutional neural network," in *Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*. ACM, 2016, pp. 26–35.

[20] K. H. Kim, S. Gaba, D. Wheeler, J. M. Cruz Albrecht, T. Hussain, N. Srinivasa, and W. Lu, "A functional hybrid memristor crossbar-array/CMOS system for data storage and neuromorphic applications," *Nano letters*, vol. 12, no. 1, pp. 389–395, 2011.

[21] B. Govoreanu, G. Kar, Y. Chen, V. Paraschiv, S. Kubicek, A. Fantini, I. Radu, L. Goux, S. Clima, R. Degraeve *et al.*, "$10 \times 10nm^2$ hf/hfo x crossbar resistive ram with excellent performance, reliability and low-energy operation," in *Electron Devices Meeting (IEDM), 2011 IEEE International*. IEEE, 2011, pp. 31–6.

[22] P.-Y. Chen, D. Kadetotad, Z. Xu, A. Mohanty, B. Lin, J. Ye, S. Vrudhula, J.-s. Seo, Y. Cao, and S. Yu, "Technology-design co-optimization of resistive cross-point array for accelerating learning algorithms on chip," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2015*. IEEE, 2015, pp. 854–859.

[23] C. Xu, D. Niu, N. Muralimanohar, R. Balasubramonian, T. Zhang, S. Yu, and Y. Xie, "Overcoming the challenges of crossbar resistive memory architectures," in *High Performance Computer Architecture (HPCA), 2015 IEEE 21st International Symposium on*. IEEE, 2015, pp. 476–488.

[24] Y. Wang, X. Li, K. Xu, F. Ren, and H. Yu, "Data-driven sampling matrix boolean optimization for energy-efficient biomedical signal acquisition by compressive sensing," *IEEE Transactions on Biomedical Circuits and Systems*, 2016.

[25] A. Vedaldi and K. Lenc, "Matconvnet: Convolutional neural networks for matlab," in *Proceedings of the 23rd ACM international conference on Multimedia*. ACM, 2015, pp. 689–692.

[26] Y. Shang, W. Fei, and H. Yu, "Analysis and modeling of internal state variables for dynamic effects of nonvolatile memory devices," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 59, no. 9, pp. 1906–1918, 2012.

[27] W. Fei, H. Yu, W. Zhang, and K. S. Yeo, "Design exploration of hybrid cmos and memristor circuit by new modified nodal analysis," *IEEE Transactions on very large scale integration (VLSI) systems*, vol. 20, no. 6, pp. 1012–1025, 2012.

[28] B. Goll and H. Zimmermann, "A 65nm cmos comparator with modified latch to achieve 7ghz/1.3 mw at 1.2v and 700mhz/47μw at 0.6v," in *2009 IEEE International Solid-State Circuits Conference (ISSCC)*. IEEE, 2009, pp. 328–329.

[29] Y.-H. Chen, T. Krishna, J. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," in *2016 IEEE International Solid-State Circuits Conference (ISSCC)*. IEEE, 2016, pp. 262–263.

[30] G. E. Hinton and R. R. Salakhutdinov, "Replicated softmax: an undirected topic model," in *Advances in neural information processing systems*, 2009, pp. 1607–1614.