Switched by Input: Power Efficient Structure for RRAM-based Convolutional Neural Network

Lixue Xia, Tianqi Tang, Wenqin Huangfu, Ming Cheng, Xiling Yin, Boxun Li,
Yu Wang, Huazhong Yang
Dept. of E.E., Tsinghua National Laboratory for Information Science and Technology (TNList),
Tsinghua University, Beijing, China
e-mail: yu-wang@mail.tsinghua.edu.cn

ABSTRACT

Convolutional Neural Network (CNN) is a powerful technique widely used in computer vision area, which also demands much more computations and memory resources than traditional solutions. The emerging metal-oxide resistive random-access memory (RRAM) and RRAM crossbar have shown great potential on neuromorphic applications with high energy efficiency. However, the interfaces between analog RRAM crossbars and digital peripheral functions, namely Analog-to-Digital Converters (ADCs) and Digital-to-Analog Converters (DACs), consume most of the area and energy of RRAM-based CNN design due to the large amount of intermediate data in CNN. In this paper, we propose an energy efficient structure for RRAM-based CNN. Based on the analysis of data distribution, a quantization method is proposed to transfer the intermediate data into 1 bit and eliminate DACs. An energy efficient structure using input data as selection signals is proposed to reduce the ADC cost for merging results of multiple crossbars. The experimental results show that the proposed method and structure can save 80% area and more than 95% energy while maintaining the same or comparable classification accuracy of CNN on MNIST.

1. INTRODUCTION

Convolutional Neural Network (CNN) is a powerful and widely used technique in computer vision area. However, CNN-based methods demand much computation and memory resource [1], and therefore CNNs are hard to be directly integrated into embedded system with limited battery resource such as smart phones and robots. As a result, an energy efficient CNN accelerator, especially on embedded system, is highly demanded.

A number of CNN accelerators have been proposed on CMOS-based platforms, such as FPGA and GPU. However, state-of-the-art CMOS-based designs still cost 10W to 20W power consumption, which can not be afforded by most embedded applications [2]. As a result, many researchers start to look for new architectures and technologies to obtain further performance and efficiency gains [3].

The emerging metal-oxide resistive random-access memory (RRAM) device provides promising solutions to boost the energy efficiency [4]. RRAM devices are able to support a large number of signal connections within a small footprint by taking advantage of the ultra-integration density. Furthermore, RRAM devices can be used to build resistive cross-point structure [5], which is also known as the RRAM crossbar. RRAM crossbar can naturally transfer the weighted combination of input signals to output voltages and realize the matrix-vector multiplication with incredible energy efficiency [6]. Considering that convolution functions in CNNs can also be regarded as multiple vector-vector multiplications (or matrix-vector multiplication), it is a promising approach that using RRAM to improve the energy efficiency of CNN for embedded applications.

DAC '16, June 05-09, 2016, Austin, TX, USA
© 2016 ACM. ISBN 978-1-4503-4236-0/16/06...$15.00
DOI: http://dx.doi.org/10.1145/2897937.2898101

The interfaces between RRAM and peripheral circuits bring new challenge. Since RRAM is used as an analog computing device, the large amounts of analog intermediate results are difficult to be directly stored, while some other functions in CNN like max pooling are difficult to implement in analog circuits. Therefore, RRAM crossbar needs additional interfaces like Analog-to-Digital Converters (ADCs) and Digital-to-Analog Converters (DACs) to communicate with digital peripheral circuits. However, the results of a 4-layer RRAM-based CNN demonstrate that ADCs and DACs cost more than 98% of the area and power consumption. As a result, the interfaces cost much more area and energy consumptions than the RRAM-crossbars, which becomes a new bottleneck of RRAM-based CNN.

In this paper, we propose an energy efficient structure for RRAM-based CNN. We quantize the intermediate data between layers into 1-bit to eliminate DACs, and use the quantized input data as selection signals to reduce the additional ADCs needed when merging the results of multiple RRAM crossbars. The contributions of this paper include:

1. We propose a greedy algorithm to quantize the intermediate data into 1-bit based on the analysis of intermediate data distribution, which reduces the DAC overhead.
2. We propose SEI, a novel structure where the RRAM crossbar weights are SElected by 1-bit Input, which has an extra port to provide the common information of weights. SEI can use single RRAM crossbar to process the computation of signed high-precision weights without merging, which reduces the ADC overhead.
3. We propose a structure to split large matrix without ADCs. A matrix homogenization method and a dynamic threshold structure implemented by SEI are proposed to compensate the accuracy reduction caused by matrix splitting.
4. In the case study of MNIST dataset on three kinds of CNNs, the proposed algorithm and structure can save 80% area and more than 95% energy consumption while maintaining the same or comparable classification accuracy of CNN. The proposed structure achieves high energy efficiency at more than 2000 GOPs/J (Giga Operations per J).

2. PRELIMINARIES AND MOTIVATION

2.1 CNN

A typical CNN consists of a number of Convolutional (Conv) layers and Fully-Connected (FC) layers that run sequentially. Generally, a Conv layer can further contain three main functions: convolution kernel, non-linear neuron, and spatial pooling.

Conv Kernel can be expressed as in Equ. (1):

$$f_o(x,y,z) = \sum_{i=0}^{S-1}\sum_{j=0}^{S-1}\sum_{k=1}^{I} f_i(x+i,y+j,k)c_z(i,j,k) \qquad (1)$$

where the 3D matrix F_i and F_o respectively represent the input and output feature map denoted by x, y, and z; C_z is one Conv kernel with the size of $S \times S \times I$; and k is the channel number of the Conv kernel and the input feature map.

Non-linear Neuron is attached after the Conv kernel which makes one-by-one mapping. Rectified Linear Unit (ReLU) function is the most common one, i.e. $h = \mathrm{ReLU}(g) = \max(g, 0)$. **Spatial Pooling** merges the neighbor area of input feature map, which chooses the maximum value of input blocks.

Fully-Connected Layers are the final layers that all inputs and outputs are connected by weights like Artificial Neural Network (ANN). Generally, the function of FC layer can be regarded as:

$$output_i = f(\sum_j w_{i,j} \times input_j + b_i) \tag{2}$$

where $\overrightarrow{input} = \{input_1, input_2, ..., input_n\}$ is the input vector of layer denoted by j, $\overrightarrow{output} = \{output_1, output_2, ..., output_m\}$ is the output vector of layer denoted by i. $\vec{b}$ is the bias vector that is only used in FC layer, and $W = (w_{i,j})_{m \times n}$ is the weight matrix. $f(x)$ represents the non-linear function like ReLU function mentioned.

2.2 RRAM Device and RRAM-based CNN

An RRAM device is a passive two-port element with multiple variable resistance states, and multiple devices can be used to build the crossbar structure as shown in Fig. 2(a). If the "matrix" is stored by the conductivity of the RRAM devices and the "vector" by the input voltage signals, the RRAM crossbar is able to perform analog matrix-vector multiplication (or vector-vector inner product). The relationship between the input and output voltage can be expressed as in Equ. (3) [3]:

$$i_{out,k} = \sum_{j=1}^{N} g_{k,j} \cdot v_{in,j} \tag{3}$$

where $\vec{v}_{in}$ is the input voltage (denoted by $j = 1, 2, ..., N$), $\vec{i}_{out}$ is the output current (denoted by $k = 1, 2, ..., M$), and $g_{k,j}$ is the conductance of RRAM device representing the matrix data. Taking advantage of the natural "multiplication and merging" function of the crossbar structure, the RRAM crossbar implements high efficiency matrix-vector multiplication in analog mode.

The operation of FC layers shown in Equ. (2) mainly contains a matrix-vector multiplication operation, which can be efficiently implemented on RRAM [3]. Since the RRAM can only implement positive resistance levels, researchers use two crossbars to process the computation of positive weights and negative weights separately and merge the results by additional ADCs and subtractors [4]. The Conv kernels in Conv layers are multiple vector-vector multiplication with the same length, which can also be regarded as matrix-vector multiplication. For example, for the Conv layer containing 64 kernels in $3 \times 3 \times 3$ size, we can use 27×64 RRAM crossbar to store all the 64 kernels, where each RRAM column stores the weights of a specific $3 \times 3 \times 3$ Conv kernel. Therefore, RRAM can implement the main functions of Conv kernels and FC layers with high speed, small area, and low power.

2.3 Motivation

When the depth of CNN and the size of pictures increase, there are huge amounts of intermediate results need to be buffered. Take VGG-19 [7] as an example, there are totally 3×10^7 pieces of intermediate data for processing single picture. Without any buffer, all the 10^9 RRAM cells of all layers need to work simultaneously until the final outputs of the whole CNN are obtained. Therefore, directly implementing all the functions in analog circuits without buffers is not practical considering both the data scale, telegraph noise, and the driving ability of crossbar input [8].

Since the analog output signals are difficult to be directly stored, while some other functions in CNN like max pooling are difficult to implement in analog circuits, the ADCs and DACs are used as interfaces between RRAM crossbar and peripheral digital functions, as Fig 2 (b) shows. Fig. 1 shows the power and area consumption breakdowns for a CNN tested in MNIST handwritten database [9] containing 2 Conv layers and 1 FC layer shown as Network 1 in Table 2. The results demonstrate that ADCs and DACs cost more than 98% of the area and power consumption of RRAM-based CNN even if the crossbar size is 512×512, which becomes a new bottleneck of RRAM-based CNN.

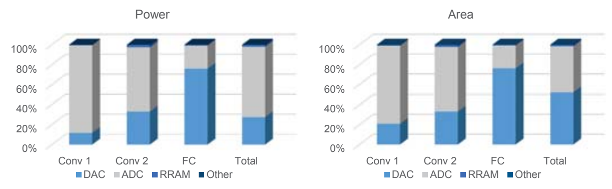

Figure 1: Power and area consumption for a 4-layer CNN with 8-bit data precision.

Table 1: Data Distribution of 5 Conv Layers in CaffeNet [12]

Range of Normalized Data	0∼1/16	1/16∼1/8	1/8∼1/4	1/4∼1
Layer 1	95.18%	3.71%	1.00%	0.11%
Layer 2	98.65%	1.20%	0.14%	0.01%
Layer 3	96.63%	2.90%	0.45%	0.02%
Layer 4	93.51%	5.10%	1.31%	0.08%
Layer 5	98.38%	1.25%	0.34%	0.03%
All Layers	98.63%	1.20%	0.16%	0.01%

2.4 Related Work on Neural Network Quantization

If the intermediate data can be quantized into 1-bit value, we can directly use the each intermediate 0/1 signal as an analog input for RRAM crossbar. Kim's work shows that a binary neural network with 1-bit intermediate data and 1-bit weights can be trained by noisy propagation [10]. However, current results only demonstrate its feasibility on full-connected neural network. Moreover, the "tanh" neuron is used in Kim's work and it might be difficult to promote this work to deeper network for more difficult tasks. Johannes' work quantizes intermediate data of the Conv layers by thresholding the output of each Conv layer into binary mode [11]. The dynamic thresholds of each layer are not fixed and are determined through analysing the distribution of intermediate data of each class, where the process of k-means clustering is introduced. In our work, it is found that it might be not necessary to cluster the intermediate data; instead, a direct robust searching method, which minimizes the quantization error, is proposed. Moreover, it can be deduced that the "sigmoid" neuron is used in Johannes's work for historical reason. In our work, the "relu" neuron is used and it might be easier to promote this work to networks with deeper layers and more complex structure.

3. SOFTWARE QUANTIZATION TO ELIMINATE DACS

3.1 Data analysis and Quantization Method

To further consider the quantization of intermediate data between Conv Layers, we analyze the distribution of intermediate data. Table 1 shows the distribution of intermediate data of 5 Conv layers in CaffeNet [12] normalized by the maximum value of each layer. We find that more than 85% of the output data of Conv layers are zeros, many other data are around zeros, and only less than 10% data are at high value. Based on this observation, we propose an algorithm to quantize all the intermediate data of CNN into 1-bit value, as shown in Algm. 1. The operation of threshold optimization contains two steps: weight re-scaling and threshold searching.

Weight Re-scaling: The intermediate outputs distribute in large range. Take the CaffeNet [12] for example, the output range of 5 Conv layers are respectively [0-2048], [0-4096], [0-4096], [0-1024], [0-1024]. To search the threshold of different layers using the same search step, the weight should be firstly re-scaled to keep the intermediate outputs of the current layer in the scale of [0-1]. Specifically, the weight is divided by the maximum output of the corresponding layer. In addition, the weight scaling without numeral precision loss does not change the classification result of CNN. Weight re-scaling only changes the range of the intermediate result of current and deeper layer, while maintaining the order of magnitude of all the output channels.

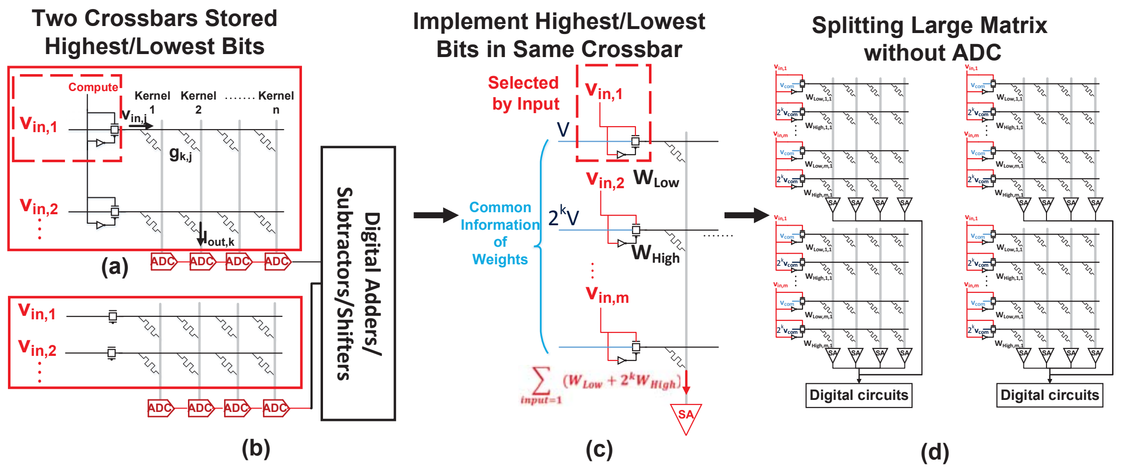
Two Crossbars Stored Highest/Lowest Bits
Compute
Kernel 1
Kernel 2
Kernel n
$V_{in,1}$
$V_{in,j}$
$g_{k,j}$
$V_{in,2}$
$I_{out,k}$
ADC
(a)
Digital Adders/ Subtractors/Shifters
(b)
Implement Highest/Lowest Bits in Same Crossbar
Selected by Input
$V_{in,1}$
V
Common Information of Weights
$V_{in,2}$
W_{Low}
2^kV
W_{High}
$V_{in,m}$
$\sum_{input=1} (W_{Low} + 2^k W_{High})$
SA
(c)
Splitting Large Matrix without ADC
Digital circuits
(d)

Figure 2: (a) Traditional circuit of RRAM crossbar and input peripheral circuits (b) ADC-based merging method (c) Proposed SEI structure using input to select weights (d) Using SEI to merge results of small crossbars without ADC.

Table 2: Experiment Setup

Layers	Network 1	Network 2	Network 3
Input Layer	28×28	28×28	28×28
Conv Layer 1	12 kernels sized of 5×5	4 kernels sized of 3×3	6 kernels sized of 3×3
Weight Matrix 1	25×12	9×4	9×6
Pooling	2×2	2×2	2×2
Conv Layer 2	64 kernels sized of 5×5	8 kernels sized of 3×3	12 kernels sized of 3×3
Weight Matrix 2	300×64	36×8	54×12
Pooling	2×2	2×2	2×2
FC Layer	1024×10	200×10	300×10
Complexity (GOPs)	0.006	0.00016	0.0003

Table 3: Error Rate of Quantization Method on MNIST

Network	1	2	3
Before Quantization	0.93%	2.88%	1.53%
After Quantization	1.63%	3.42%	2.07%

Layer-by-Layer Greedy Strategy: Since the 1-bit quantization also influences the distribution and range of following layers, we use a greedy algorithm which optimize the threshold layer by layer. This method searches different threshold for each layer because the data distribution differs, which can reduce the accuracy loss of quantization.

Threshold Searching: The threshold of a specific layer is searched by a brute-force method. According to our experiences, the intermediate output of each Conv layer has a long-tail distribution, where most of the data are exactly zero or close to zero, as Table 1 shows. Therefore, the threshold is searched from 0 to 0.1 because the optimized threshold is usually much smaller than 0.1.

To avoid over-fitting problem on threshold optimization, we use the 60,000 samples in $Training\ Set$ of CNN to optimize the quantization threshold, while the experimental results are tested in the 10,000 samples in $Test\ Set$. Three CNNs in Table 2 are tested on MNIST handwritten dataset [9]. All the networks have a similar data distribution with CaffeNet, where the intermediate data contains more than 95% values around zero. The results of Networks are shown in Table 3. The classification accuracy only reduces less than 1% after 1-bit quantization. Quantizing the intermediate data into 1-bit value makes the intermediate data can be directly processed by RRAM crossbars without conversion, which also provides the potential to eliminate ADCs of RRAM crossbar. Furthermore, all generally used non-linear neurons like sigmoid and ReLU functions are monotonically increasing functions, which means the neuron function can also be merged into the SA by setting a corresponding reference. In addition, quantizing the data after max pooling is equivalent to quantizing before pooling with the same threshold, we can process this quantization before pooling, and therefore the max pooling function degenerates into an ąřORąś function of 1-bit inputs.

3.2 Discussion About Input Layer

Algorithm 1: Threshold Searching Algorithm

```
Input: $CNN$, $TrainingSet$, $Thres_{min}$, $SearchStep$, $Thres_{max}$
Output: $Threshold$
1  Initial $Threshold$ as Empty Array;
2  for $L = 1 : CNN.LayerAmount$ do
3      Using the quantization results of front layers to obtain outputs of Layer $L$:$Output(L)$ = Feedforward($CNN$,$TrainingSet$,$Threshold$);
4      $CNN.Weight(L) = CNN.Weight(L)$ / max($Output(L)$) ;
5      $Accuracy_{max}$ = 0;
6      for $Thres_{temp} = Tresh_{min} : SearchStep : Thres_{max}$ do
7          $Accuracy_{temp}$ = Feedforward($CNN$,$TrainingSet$,$Threshold$,$Thres_{temp}$);
8          if $Accuracy_{temp} > Accuracy_{max}$ then
9              $Threshold(L) = Thres_{temp}$;
10         end
11     end
12 end
13 return $Threshold$
```

Although the quantization method can reduce the precision of intermediate data, the input pictures still need high precision to provide enough information to CNN. As a result, the input layer of whole CNN still needs DACs to transfer input pictures into analog signals for RRAM-based design. In this paper, we directly use DAC-based design because input layer takes a very small part of CNN. For example, the input layer DACs cost about 3% energy consumption and only 1% area of the whole chip in the 4-layer CNNs mentioned in Table 2, and the partition will further decrease when the scale of CNN grows deeper and larger.

4. HARDWARE STRUCTURE DESIGN TO ELIMINATE ADCS

Software 1-bit quantization can reduce DACs of RRAM-based CNN because the intermediate data can be directly used as input signals for RRAM crossbars. However, the ADCs are still demanded when the output signals of RRAM crossbars need to be merged with other results instead of directly quantized by threshold processing. Therefore in this section, we propose a novel structure design to eliminate the ADCs for merging.

Specifically, there are three kinds of situation that we need to merge results of multiple crossbars before threshold processing. Firstly, the weights of CNNs usually need at least 8-bit precision [7]. However, the state-of-the-art RRAM devices can only support 4 to 6 bit of resistance levels [13], which means we need multiple crossbars to store a part of bits of weight in each crossbar [14] and merge the results together, as shown in Fig. 2(b). Secondly, the weights in CNNs use signed values, but the

RRAM device can not achieve negative resistance. Therefore, we need two crossbars to store the positive and negative weights respectively [4]. Finally, Limited by the IR-drop phenomenon with other fabrication problems, state-of-the-art RRAM crossbars only achieve 512×512 size [15]. Nevertheless, the Conv kernels of deep layers and weight matrixes in FC layers need to perform large-scale matrix-vector multiplication, which cannot be directly implemented in single crossbar [7]. As a result, we need to split the large weight matrix into multiple blocks and implemented on multiple crossbars.

4.1 Implement Signed High-precision Weights in Single Crossbar

Merging problem of high/low-precision-bits crossbars and positive/negative crossbars have some similar characteristics. The weights stored in each crossbar share some common information, like the sign and precision-bits. Therefore, if we can introduce the common information into the computation structure, the weights can be implemented in single crossbar without any additional merging function.

Motivated by this consideration, we view back onto the function of Conv layers and FC layers to explore the potential advantages brought by 1-bit quantization. Equ. (2) shows the layer function of FC layers, which also represents the function of Conv Layers. Specifically, $\overrightarrow{input}$ is a specific feature map, W contains the Conv kernels where $\vec{w_j} = w_{1,j}, w_{2,j}, ..., w_{m,j}$ represents a specific kernel like the $\vec{c}_z$ in Equ. (1), and $\overrightarrow{output}$ is one 3D pixel of output feature map before pooling layer, namely an $1 \times 1 \times k$ matrix if the amount of kernels is k.

After 1-bit quantization of intermediate data, the function can be simplified into:

$$output_i = \begin{cases} 1 \quad if(\sum\limits_{input_j=1} w_{i,j} + b_i > Threshold) \\ 0 \quad if(\sum\limits_{input_j=1} w_{i,j} + b_i <= Threshold) \end{cases} \tag{4}$$

From this view, we find that the quantized 1-bit input data of Conv and FC layers can be regarded as selection signals only to control the accumulation of weights. If we can transfer the input data onto some control ports of RRAM crossbar, the original "input" ports can introduce extra common information of the weights in the same row, which just meets our intention.

Based on the above analysis, we propose SElected by Iput (SEI) structure, which uses the 1-bit input data of RRAM-based computing as selection signals to drive the transmission gates of RRAM crossbar. Specifically, the complete structure of a traditional RRAM crossbar with input peripheral circuits is shown in Fig. 3(a). A decoder is used to choose the column and row of crossbar and to select a specific cell to be written or be verified before performing computation. The decoder generates 1-bit selection singles to turn on the corresponding transmission gates, and an OR gate is used to turn on all transmission gates for computation. In SEI structure, instead of turning on all cells, we use 1-bit input signals to select the input transmission gates. Therefore, an extra signal can be mapped on the original "input" lines as shown in Fig. 2(c). The additional ports can share some common information of the weights in the same row, like the precision bias and positive/negative sign of weights.

From a more general view, if the results of crossbars are needed to be merged by a weighted sum before the threshold processing, the ADC-based merging method can be represented by:

$$A_1(\sum_{in_j=1} w_{(1)j} + b_1) + A_2(\sum_{in_j=1} w_{(2)j} + b_2) + ... > Thres \tag{5}$$

where $w_{(k)}$ are the vectors of weight matrix to generate the same output feature, and A_k is the "weight" (like 2^4 for higher-4-bit-precision crossbar) of the results of the kth crossbar. Taking the extra signal port into account, we can rewrite Equ. (5) into:

$$\sum_{in_j=1} \sum_{k=1}^{K} (A_k w_{(k)j}) > Thres - B \tag{6}$$

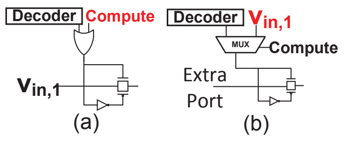

Figure 3: (a) Traditional computing decoder (b) SEI decoder.

where $B = \sum_{k=1}^{K} A_k b_k$ is the total bias signal of K crossbars, which can be merged into threshold because it is independent with weight and input data. Equ. (6) shows that the weighting operation of results can be taken into the matrix-vector multiplication by the extra port provided by SEI. In our situation, A_k can be chosen as 1 and 2^4 to implement high-precision weights, or be chosen as 1 and -1 to implement signed weights. As a result, we can use two RRAM cells in the same column to store the highest bits and the lowest bits of a specific weight, and use different bias voltages in the input ports to distinguish them and implement the "shift and add" function in the computing phase, as shown in Fig. 2 (c). The signed weight can also be implemented by two cells in the same column using positive/negative "input" signals separately. Therefore, SEI structure can use only one crossbar to implement the computing of signed and high-precision weight matrix without additional merging, and reduce the large power and area overhead consumed by the interfaces.

4.2 Dynamic Threshold

For some unipolar devices or the devices with unsymmetrical bipolar performance [16], using negative "input" to represent negative weight is not available. To implement the signed wights in single crossbar without introducing negative inputs, we propose a dynamic threshold structure based on SEI, which can naturally implement a linear mapping to transfer all the signed weights into positive values. Specifically, we can regard the whole range of positive and negative weights as a large weight interval and map it onto RRAM device. For original CNN, the function after linear transformation is:

$$out = f(\sum_j k(w_j^* - w_0) \times in_j + b) \tag{7}$$

where w^* is the weights stored in RRAM, and k and w_0 are the parameters for linear transformation. In this method, we introduce a dynamic bias depending on the input data in_j, while the influence of the dynamic bias over non-linear function is difficult to solve. But after 1-bit quantization, the function with linear transformation can be simplified into:

$$(\sum_{in_j=1} k(w_j^* - w_0) + b) > Thres \tag{8}$$

Therefore we only need to solve a biased threshold processing:

$$k \sum_{in_j=1} w_j^* + b > Thres + k \sum_{in_j=1} w_0 \tag{9}$$

Although the slope k of linear transformation is easy to be implemented by extra port of SEI structure, the bias of threshold still depends on the input data. To deal with this problem, we use an additional RRAM column, which is also selected by input data, to implement the dynamic threshold, as Fig. 4 shows. If we map k on the extra "input" port and store the bias w_0 into the cells of additional RRAM column, the output signal of the rightmost column is the dynamic part of threshold $k \sum_{in_j=1} w_0$. Furthermore, we can store the $Thres$ of Equ. (9) into the bottom right corner of crossbar, and therefore the output of rightmost column is exactly the threshold to be compared with the left rows' results. As a result, we can use SEI structure with dynamic threshold processing and to perform the linear transformation of weights.

4.3 Splitting Large Matrix without ADCs

Different from the merging of higher/lower weights and positive/negative weights, the crossbars that are split from large matrix have little common relationship with each other. Specifically, the different cells in a row vector work for different vector-vector multiplications and they are independent with each other. Therefore, we only consider the column splitting because the results of

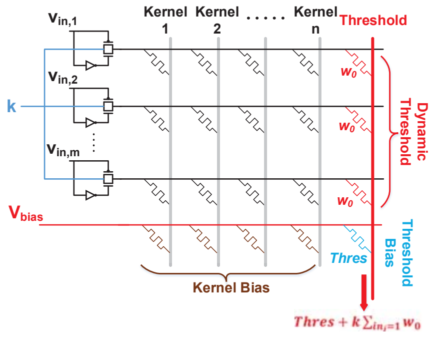

Figure 4: Dynamic Threshold.

different parts need to be added together, which introduces additional ADCs.

Considering the function of non-linear neuron has been simplified into a threshold processing, we can directly divide the original threshold into multiple parts for the crossbars, like using $Thres/3$ as the threshold for 3 individual crossbars, as Fig. 2 (d) shows. We define a new digital threshold for the sum of sub-matrix results, and use some digital peripheral circuits to process the 1-bit out signals of sub-matrix. However, the accuracy of CNN reduces a lot because both the weight and input features are non-uniform. For example, if the result of first part is big enough to fire the threshold processing while the results of other two parts are very small, the final 3 outputs contain two 0 and single 1, which will be recognized as 0 and cause error.

The above error is mainly caused by the randomness of both input data and weights. In this paper, we propose an off-line homogenization method to enhance the priori knowledge of the weight and reduce randomness, and an on-line dynamic threshold method to compensate the input data randomness based on the posteriori distribution obtained by dynamic threshold structure of SEI shown in Fig. 4.

Enhancing priori knowledge of weight matix: Considering that all the weight matrixes are unchanged during computing, we can adjust the distribution of weights between sub-matrixes to a more uniform one to enhance the priori knowledge before mapping matrix onto RRAM crossbars. As a result, we want to homogenize the sum of sub-columns to be merged between different crossbars. The target function is defined as the total Euclidean distance between average vectors of sub-matrixes:

$$dist = \sum_{i \neq j}^{K} (\vec{a}_i - \vec{a}_j) \qquad (10)$$

where $\vec{a}_i$ is the $1 \times n$ mean value vector of the $m \times n$ sub-matrix w_i averaged column by column.

Specifically, we only need to adjust the combination of rows in each sub-matrix. This problem can be regarded as dividing multiple weight rows into K parts and minimize the distance of average value row-vector of K sub-matrixes. The optimization can be divided into multiple Knapsack problems, which is a NPC problem. The brute-force method is acceptable because this optimization only needs to be processed once before mapping the CNN weights onto RRAM crossbar. We also use a heuristic algorithm to find a local optimum because we do not need the exactly optimal solution for an approximate application. We use a genetic algorithm to iteratively optimize the combination of row-vectors by randomly exchange the position of two vectors. Results shows that for fine-trained CNN models, the total distance can be reduced about 80% to 90% compared with directly splitting the matrix by natural order. The experimental results also show that the whole CNN can only obtain 54.21% of classification accuracy using random order. Taking advantage of the proposed reorder operation for weight homogenization, the accuracy rises back to 98.22%, which illustrates the effect of weight matrix homogenization.

Compensating posteriori knowledge of input data: The dynamic threshold structure discussed in Section IV.B can introduce the posteriori information of current input data to support the judgement, which reduce the influence of input randomness.

Table 4: Error Rate of Proposed Methods on Network 1

Max Crossbar Size	512	256
Original CNN	0.93%	0.93%
Quantization	1.63%	1.63%
Random Order Splitting	3.90 - 45.89%	4.44 - 49.03%
Matrix Homogenization	1.78%	2.29%
Dynamic Threshold	1.52%	1.82%

Specifically, the threshold processing results of different sub-matrixes are more fair if we give the sub-matrix with more low-value inputs (namely "0") a lower threshold. Therefore, some "0,0,1" situations as the example mentioned above can be compensated to "0,1,1" or even "1,1,1", leading to a correct judgement. We use the 60,000 samples in $Training\ Set$ to optimize the interval of dynamic threshold, while the experimental results are tested in the 10,000 samples in $Test\ Set$. The results show that dynamic threshold method can further reduce about 0.4% of error rate, and the final accuracy may even better than the no-splitting but static-threshold results.

In addition, it is obvious that the proposed methods can not completely solve all extreme situations after matrix splitting. Nevertheless, the total classification accuracy of large amounts of samples can be improved, which supports the performance of splitting structure without ADCs and obtains large energy saving.

5. EXPERIMENTAL RESULTS

5.1 Experiment Setup

In this section, three 4-layer CNNs with different kernel sizes and different kernel amounts are demonstrated on MNIST handwriting dataset [9] as a case study to show the effectiveness of the proposed method. The detailed configurations about the CNNs are shown in Table 2. The precision of weight matrix is 8-bit while the precision of RRAM device is 4-bit. The area and power data of analog peripheral circuits and RRAM devices are taken from [17–19]. The energy costs of obtaining picture data from memory and other digital functions are taken from [20]. For the accuracy emulation, an 4-bit RRAM device model packed in Verilog-A [21] is used to build up the SPICE-level crossbar array. We also show the performance of CNNs on FPGA [2] and Nvidia K40 GPU platform for comparison.

In addition, to provide a fair comparison, any size of crossbar can be used if it is smaller than the maximum size limited by experiment configuration. Therefore the amount and size of crossbars are different for different methods. Take the Network 1 as an example, the Covn Layer 2 needs 300×64 matrix-vector multiplication using signed 8-bit weights. Given that the maximum RRAM crossbar size is $512\times$ 512 with 4-bit precision, the ADC-based method implements the matrix in 300×64 crossbar but demands total 4 crossbars. When using SEI structure, despite the fact that we can use 4 cells to implement a weight in the same crossbar, we still need three 400×64 crossbars to implement the huge 1200×64 RRAM array.

5.2 Results of Accuracy

Table 4 shows the detailed accuracy results of the proposed methods and structure tested on Network 1. The results show that the error rate of SEI structure only increases about 1% compared with the original software CNN. To illustrate the importance of matrix homogenization, we randomly chose 500 orders (combination methods) of row vectors for matrix splitting. It can be seen from the results that the error rate of random order varies in a large range up to around 50%, but can be reduced back to less than 2% by matrix homogenization. This result shows that the proposed homogenization method is necessary to reduce the randomness of weight matrix and maintain the performance of CNN after splitting, which also illustrates that the distance definition is reasonable for our application.

5.3 Energy and Area Saving

Since each kernel is used multiple times in the procession of one picture, we can use buffer amounts to trade-off the power with time. But if we don't take the buffers' overhead into consideration, the energy consumption maintains similar. Therefore,

Table 5: Result of Proposed Method Using 4-bit RRAM Device

Network	Data Bits	Crossbar Structure	Maximum Crossbar Size	Error Rate (%)	Energy (uJ/pic)	Saving (%)	Area Saving (%)
Network 1 5 × 5 kernel 6M Operations	8	DAC+ADC	512×512	0.93	74.25	-	-
	1	1-bit-Input + ADC		1.63	62.31	16.08	47.59
	1	SEI		1.52	2.58	96.52	86.57
	1	DAC +ADC	256×256	0.93	93.75	-	-
	1	1-bit-Input +ADC		1.63	81.80	32.74	36.81
	1	SEI		1.82	2.68	97.15	80.76
Network 2 3 × 3 kernel 0.16M Operations	8	DAC+ADC	512×512	2.88	12.15	-	-
	1	1-bit-Input +ADC		3.42	10.45	13.97	56.31
	1	SEI		3.46	0.68	94.37	78.50
Network 3 3 × 3 kernel 0.3M Operations	8	DAC+ADC	512×512	1.53	17.77	-	-
	1	1-bit-Input +ADC		2.07	292.01	15.22	53.35
	1	SEI		2.07	0.73	95.89	74.35

we use the energy consumption of single picture to evaluate the energy efficiency. As for area reduction, considering that the final layout area after routing and other technological processes is difficult to predict, we only estimate the minimum area cost of all analog and digital modules, and use the design which reuses the kernels (crossbars) for multiple feature maps as the baseline for comparison.

The experimental results in Table 5 shows that SEI-based design can save more than 95% of energy consumption compared with original ADC/DAC-based design, and also saves more than 90% compared with the ADC-based merging structure after data quantization. As for area part, all SEI-based designs obtain 74% to 86% savings compared with orginal structure. The SEI-based design can obtain high energy efficiency at more than 2000 GOPs/J, which is about 2 orders of magnitude higher than state-of-the-art FPGA [2] and GPU implementations. The energy efficiency gains and area saving further increase if we have to use smaller crossbars and merge more times, or the kernel size is small enough that both positive/negative and higher/lower-precision weights can be stored into the same crossbar when using SEI.

6. CONCLUSIONS

In this paper, we propose an power efficient structure for RRAM-based CNN. Based on the analysis of the distribution of intermediate data, a quantization method is proposed to quantize intermediate data into 1 bit and eliminate DACs. A power efficient structure using inputs as selection signals is proposed to eliminate ADCs for merging. We further propose the splitting method of a large matrix without ADCs. A matrix homogenization method and a dynamic threshold structure are proposed to compensate the accuracy reduction of matrix slitting. Experimental results show that the proposed algorithm and structure can save 80% area and more than 95% energy consumption while maintaining the same or comparable classification accuracy of CNN.

In the future, we will further analyze the register buffer design in Conv layers, and research the complete design optimization flow for RRAM-based CNN considering the non-ideal factors of RRAM and circuit. We will also use the proposed structure to support other applications using 1-bit data like RRAM-based Spiking Neural Networks (SNNs) [22].

7. ACKNOWLEDGEMENTS

This work was supported by 973 project 2013CB329000, National Science and Technology Major Project (2011ZX03003-003-01, 2013ZX03003013-003) and National Natural Science Foundation of China (No.61373026, 61261160501, 61271269), The Importation and Development of High-Caliber Talents Project of Beijing Municipal Institutions, and Tsinghua University Initiative Scientific Research Program.

8. REFERENCES

[1] J. Qiu *et al.*, “Going deeper with embedded fpga platform for convolutional neural network,” in *ACM International Symposium on FPGA*, 2016.

[2] C. Zhang *et al.*, “Optimizing fpga-based accelerator design for deep convolutional neural networks,” in *FPGA*, 2015, pp. 161–170.

[3] M. Hu *et al.*, “Hardware realization of bsb recall function using memristor crossbar arrays,” in *DAC*, 2012, pp. 498–503.

[4] B. Li *et al.*, “Memristor-based approximated computation,” in *ISLPED*. IEEE Press, 2013, pp. 242–247.

[5] C. Xu *et al.*, “Design implications of memristor-based rram cross-point structures,” in *DATE*, 2011, pp. 1–6.

[6] L. Xia *et al.*, “MNSIM: Simulation platform for memristor-based neuromorphic computing system,” in *DATE*, 2016, pp. 469–474.

[7] K. Simonyan *et al.*, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.

[8] Y. Zhang *et al.*, “Random telegraph noise analysis in alox/woy resistive switching memories,” *Applied Physics Letters*, vol. 104, no. 10, p. 103507, 2014.

[9] Y. LeCun *et al.*, “The mnist database of handwritten digits,” 1998.

[10] M. Kim *et al.*, “Bitwise neural networks,” in *ICML workshop*, 2015.

[11] J. Fieres *et al.*, “Training convolutional networks of threshold neurons suited for low-power hardware implementation,” in *IJCNN*. IEEE, 2006, pp. 21–28.

[12] A. Krizhevsky *et al.*, “Imagenet classification with deep convolutional neural networks,” in *NIPS*, 2012, pp. 1097–1105.

[13] F. Alibart *et al.*, “High precision tuning of state for memristive devices by adaptable variation-tolerant algorithm,” *Nanotechnology*, vol. 23, no. 7, p. 075201, 2012.

[14] D. Garbin *et al.*, “Hfo2-based oxram devices as synapses for convolutional neural networks,” *TED*, vol. 62, no. 8, pp. 2494–2501, Aug 2015.

[15] M. Catanzaro *et al.*, “Reconfigurable rram for lut logic mapping: A case study for reliability enhancement,” in *SOCC*, 2012, pp. 94–99.

[16] Y. Zhang *et al.*, “Study of conduction and switching mechanisms in Al/AlOx/WOx/W resistive switching memory for multilevel applications,” *Applied Physics Letters*, vol. 102, no. 23, p. 233502, 2013.

[17] R. St Amant *et al.*, “General-purpose code acceleration with limited-precision analog computation,” in *ISCA*, 2014, pp. 505–516.

[18] W.-H. Tseng *et al.*, “A 960ms/s dac with 80db sfdr in 20nm cmos for multi-mode baseband wireless transmitter,” in *VLSI Circuits Digest of Technical Papers*, 2014, pp. 1–2.

[19] B. Li *et al.*, “Merging the interface: Power, area and accuracy co-optimization for rram crossbar-based mixed-signal computing system,” in *DAC*, 2015.

[20] S. Han *et al.*, “Learning both weights and connections for efficient neural networks,” *arXiv preprint arXiv:1506.02626*, 2015.

[21] S. Yu *et al.*, “A low energy oxide-based electronic synaptic device for neuromorphic visual systems with tolerance to device variation,” *Advanced Materials*, vol. 25, no. 12, pp. 1774–1779, 2013.

[22] T. Tang *et al.*, “Spiking neural network with rram: Can we use it for real-world application?” in *DATE*, 2015, pp. 860–865.