An On-chip Layer-wise Training Method for RRAM based Computing-in-memory Chips

Yiwen Geng, Bin Gao, Qingtian Zhang, Wenqiang Zhang, Peng Yao, Yue Xi, Yudeng Lin,
Junren Chen, Jianshi Tang, Huaqiang Wu and He Qian
Institute of Microelectronic, Beijing Innovation Center for Future Chips (ICFC), Tsinghua University,Beijing, China
Email: gaob1@tsinghua.edu.cn, zhangqt0103@tsinghua.edu.cn

***Abstract*—RRAM-based computing-in-memory (CIM) chips have shown great potentials to accelerate deep neural networks on edge devices by reducing data transfer between the memory and the computing unit. However, due to the non-ideal characteristics of RRAM, the accuracy of the neural network on the RRAM chip is usually lower than the software. Here we propose an on-chip layer-wise training (LWT) method to alleviate the adverse effect of RRAM imperfections and improve the accuracy of the chip. Using a locally validated dataset, LWT can reduce the communication between the edge and the cloud, which benefits personalized data privacy. The simulation results on the CIFAR-10 dataset show that the LWT method can improve the accuracy of VGG-16 and ResNet-18 by more than 5% and 10%, respectively, with only 25% operations and 35% buffer compared with the back-propagation method. Moreover, the pipe-LWT method is presented to improve the throughput by three times further.**

***Keywords*—RRAM, On-chip training, Computing in memory**

I. Introduction

Deep neural networks (DNNs) have shown great performance in many domains, such as automatic pilot and machine translation [1]. To promote the application of DNNs in the Internet-of-Things (IoT) era, developing new edge hardware is a good way to perform DNNs efficiently. The DNN accelerators for IoT applications like UNPU [2] have been proposed to improve energy efficiency by optimizing the data flow. However, the huge energy consumption of data transmission between the memory and the computing units limits these von Neumann chips [3]. The computing-in-memory (CIM) chips, which utilize analog computing inside the memory, offer a promising way towards efficient chips [4].

Resistive random access memory (RRAM) based CIM chips have attracted much attention due to the excellent device scalability, bidirectional analog switching behavior, and small switching energy [5]. By exploiting the fact that the conductance of the RRAM cross-point array can represent a matrix, the multiply-accumulate (MAC) operation can be performed by applying the voltages on one side of the array and reading the output currents on the other side. However, the computing precision of RRAM based CIM chips is usually affected by the device's non-ideal characteristics [6], such as variation, relaxation, nonlinear analog switching behavior, and yield. Several methods have been proposed to deal with accuracy loss induced by the non-ideal characteristics of the device. These methods are mainly divided into two categories: device engineering and system optimization. On the device side, main strategies are optimizing the device and reducing the non-ideal effects. For example, the analog switching behavior of the device can be optimized by changing the material stack [7] and the device operation scheme [8]. However, the intrinsic conductive mechanism of RRAM makes it hard to achieve an ideal device. In the system side, the off-chip [9] and on-chip [10] fine-tuning methods are two mainstream approaches to improve accuracy. The off-chip training methods, which require data and model transmission between the cloud and the edge devices, are hard to deploy into the complex environment and protect the individual privacy for IoT applications. In contrast, the on-chip methods can eliminate data transmission by training on the edge devices. However, the conventional back-propagation training algorithm is not suitable for on-chip training because it needs a large on-chip buffer space to calculate the gradient for DNN. Hence, a localized on-chip training method is important in CIM systems.

This work is supported in part by the MOST of China (2019YFB2205103) and NSFC (92064001, 92064015, 61874169).This work is supported in part by the MOST of China (2019YFB2205103) and NSFC (92064001, 92064015, 61874169)

This paper proposes an on-chip layer-wise training method to improve the accuracy of neural networks on RRAM-based CIM chips and evaluate the performance comprehensively. The main contributions are as follows:

- We propose a general layer-wise training (LWT) framework for RRAM based CIM chip and apply it to reduce the accuracy loss caused by the RRAM non-ideal factors.
- We propose an accuracy difference-aware training method for LWT to further reduce training time and energy consumption with slight accuracy loss. Furthermore, we employ a pipeline in pipe-LWT to achieve 3× data throughput.
- We evaluate the accuracy, operations, and buffer of the back-propagation and the LWT method. The experiments with VGG-16 and ResNet-18 on the CIFAR-10 dataset show that the LWT method achieves more than 5% and 10% improvement of accuracy, which is better than the back-propagation method. Further, compared to the back-propagation method, the LWT method reduces about 75% operations and 65% buffer.

The rest of the paper is organized as follows. Section II describes the proposed LWT method. Section III demonstrates the experimental results and analysis. Section IV concludes this paper.

II. Methods

A. A Compact Model of RRAM

In this work, the statistical characteristics of TiN/HfO_x/TaO/TiN RRAM in the 1K-bit array are measured to build a compact model. In the measurement, we consider read and write noise, relaxation, and analog behavior. The conductance of RRAM is divided into ten states, ranging from 2μS to 20μS. For each state, we read after mapping for 1000 devices and

calculate the mean and variance. As shown in Fig. 1(a), the middle-conductance states have a slightly larger variance than high-conductance and low-conductance states. Thus, the noise model can be generated by a gaussian distribution.

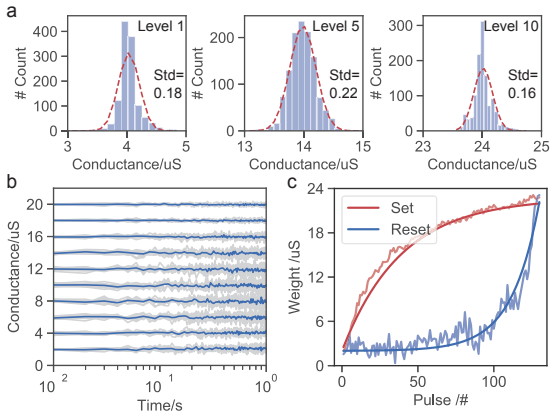

Fig. 1. Measured characteristics and compact model: variations (a), relaxation (b), fluctuations in the tuning process (c).

As shown in Fig. 1(b), the conductance drift, which is called relaxation, appears in a short period after programming. For each state, we continually read a device in one second and model the characteristics of different conductance states based on 32 RRAM devices' results. The relaxation model is described as the following equations:

$$\mathrm{Std}_{\mathrm{relax}} = (-0.0008G^2 + 0.028G - 0.0049)e^t, \quad (1)$$

where G is the conductance of the specified state and t is the time after mapping. In addition, in the large-scale array, the yield is inevitable owing to the fabrication. In this paper, we take yield, variation, and the relaxation effect into the compact model.

In the update process, ideally, the amount of weight increasing and decreasing should be linearly proportional to the number of programming pulses. However, the conductance of the RRAM device usually changes rapidly at the beginning stages of the Set and Reset process and gradually saturates. The RRAM analog behavioral model is built, as shown in (2) and (3):

$$G_{set} = 18.54 \times \left(1 - e^{-\frac{P}{0.3\times P_{max}}}\right) + 2, \quad (2)$$

$$G_{reset} = 19.08 \times e^{-\frac{P}{0.14\times P_{max}}} + 2, \quad (3)$$

where P is the number of applied pulses, the P_{max} is the number of pulses to make the conductance saturate, G_{set} and G_{reset} are the predictive conductance of the device.

B. Layer-wise Training Method

The LWT method is employed to retrain the networks, as shown in **Algorithm 1**. Firstly, the layer-wised inputs (IFM^l_{std}), the weights (W^l_{off}) and the outputs (OFM^l_{std}) are extracted from the pre-trained model, where l is the index of a layer in the networks (line1, 2). Then, the W^l_{off} are mapped on the 2T2R array as G^l_+, G^l_- (line3). For each iteration, we first calculate the on-chip output OFM^l_{on} by convolving ($G^l_+ - G^l_-$) and IFM^l_{std} (line6). Then, we calculate the loss ($loss^l$) and the residual (err^l) in layer l (line7, 8). Next, we calculate the gradient of weight (δ_{Wb}) in each batch by doing the out production between $IFM^l_{std_b}$ and err^l_b (line 9). In the next step, we update the conductance of the RRAM. If $|\delta_{W_b}|$ is larger than the threshold, we set G^l_+ and reset G^l_- when $\delta_{W_b} > 0$, otherwise we reset G^l_+ and set G^l_- (line10-16). Finally, we check the convergence of the retraining process and calculate the on-chip accuracy (line19).

Algorithm 1 On-chip layer-wise training for RRAM based computing

Input: The DNN model, Dataset, Update threshold $thres$;
Output: Optimized on-chip DNN model, Accuracy of the on-chip DNN model;
1: Training the DNN model off-chip: W^l_{off};
2: Calculate the input and output feature maps of each layer: IFM^l_{std}, OFM^l_{std};
3: Map the DNN model to the chip:$W^l_{off} \rightarrow G^l_+, G^l_-$;
4: **for** each layer l **do**
5: **repeat**
6: Calculate the on-chip output of l:$OFM^l_{on} = IFM^l_{std} * (G^l_+ - G^l_-)$;
7: Calculate the loss of l:$loss^l = L(OFM^l_{on}, OFM^l_{std})$;
8: Calculate the residual of l: $err^l = \frac{\partial L}{\partial OFM^l}$;
9: Calculate the gradient of weight δ_{W_b}: $\delta_{W_b} = \sum(IFM^l_{std_b} * err^l_b)$
10: **if** $\delta_{W_b} > thres$ **then**
11: Set G^l_+ and Reset G^l_-
12: **else if** $\delta_{W_b} < -thres$ **then**
13: Reset G^l_+ and Set G^l_-
14: **else**
15: Do nothing
16: **end if**
17: **until** ($epoch^l \geq N$)
18: **end for**
19: Calculate the on-chip accuracy;
20: **return** *On-chip model*, *On-chip accuracy*;

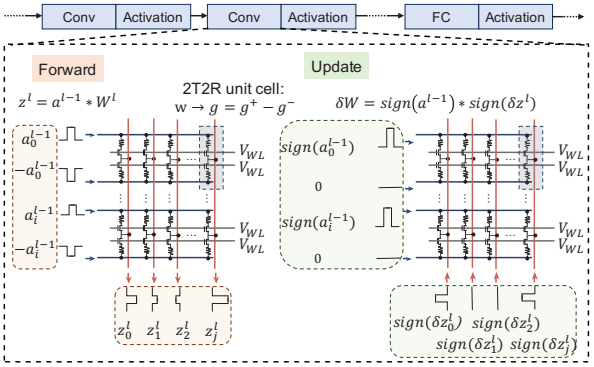

Fig. 2. The implementation of LWT method on 2T2R crossbar array.

Fig. 2 illustrates the realization of the LWT method in the circuit level. In the RRAM array, we use a 2T2R cell to represent one synaptic weight. In the inference process, according to Kirchhoff's current law, when applying opposite voltages (a_i and -a_i) to the upper RRAM (g^+) and the lower RRAM (g^-), the output current is the result of $a_i(g^+ - g^-)$. In this process, we use the amplitude of the input signal to represent the quantized feature map. In the updating process, we assign a^{l-1} and δz^l to zero, where the value is less than the threshold. Then we apply the sign of input a^{l-1}_b to the bit line and apply the sign of residual to the source line

respectively to update the conductance of the RRAM array. This method is equal to **Algorithm 1** (line 9-16) and avoids saving the weight gradient to buffers.

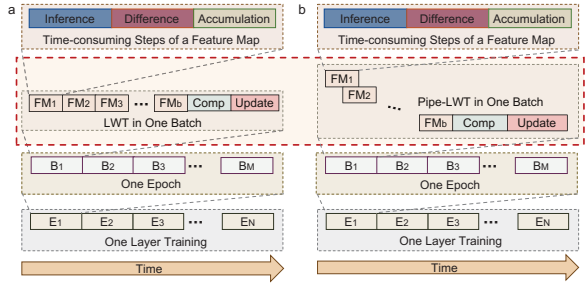

Fig. 3. The LWT method (a) and the LWT method with pipeline (b).

C. Pipeline in Training Method

In the origin LWT method, computing one batch of the feature maps sequentially leads to the overhead of cycles. To improve the throughput, we proposed the pipelined LWT (pipe-LWT) method. In the LWT method, the main time-consuming operations include the MAC operations in inference (line6), residual computing in the difference phase (line7, 8) and accumulating the residual of output (line9). These three operations are formed into atomic time steps to compute one feature map (FM_i, i is the number of a feature map among one batch). The origin LWT method sequentially calculates each feature map in a batch (B_j, j is the number of batches among one epoch), as shown in Fig. 3(a). In an epoch, B_j is conducted sequentially for M times and this process conducts for N epochs to accomplish the training for one layer.

The difference between these two methods is that the pipe-LWT method can process three feature maps in three different operations simultaneously. As shown in Fig. 3(b), the first feature map inputs the array and gets the current output. After that, the second feature map inputs to the array while the first one does the difference operation. Then, the third feature map inputs to the array while the second does the difference operation and the first does the accumulation operation. At the end of the batch, comparing and updating are conducted and enter the next iteration. In this way, the pipe-LWT method can offer fast dataflow in one batch by decoupling inference/ difference/accumulation operation.

III. Experimental Results

A. Experimental Setup

We evaluate the performance of the on-chip back-propagation method and LWT method in terms of VGG-16 and ResNet-18 on the CIFAR-10 dataset. This section presents the accuracy, buffer, operations and throughput of the origin LWT, optimized LWT (LWT-opt), and pipe-LWT algorithms. Firstly, in the experiment, we trained the VGG-16 network to 90.00% and the ResNet-18 network to 91.67%. Then we extract the parameters of the networks, including weights and feature maps. In the on-chip back-propagation method, we applied the RRAM-based updating model and trained for 20 epochs. In the LWT method, we trained for 20 epochs with activation function and mean square error function to evaluate each layer's output.

B. Results and Discussion

1) Accuracy: We compared the accuracy of the on-chip back-propagation method, LWT method and LWT-opt method. The LWT method has shown a better capability to improve accuracy. Due to the non-ideal characteristics, the recognition accuracy initially drops from 90.00% and 91.67% to 84.65% for VGG-16 and 80.70% for ResNet-18, respectively. For the on-chip back-propagation method, each network was trained with 20 epochs and the accuracy has been improved to 89.44% and 88.88%. For the LWT method, each layer was trained with 20 epochs. As we can see in Fig. 4, the trend of the recovered recognition accuracy curve with the LWT method is fast in the earlier layers and becomes saturated after C4.1 and C2.2.1 in VGG-16 and ResNet-18, respectively. The recovered accuracy of the LWT method is 89.83% and 91.13%, which are close to the origin accuracy of 90.00% and 91.67%. Compared to the back-propagation method, the LWT method achieves 0.39% and 2.25% improvement of accuracy. This improvement of the LWT method is mainly due to eliminating error accumulation caused by the non-ideal characteristics of devices in the back-propagation.

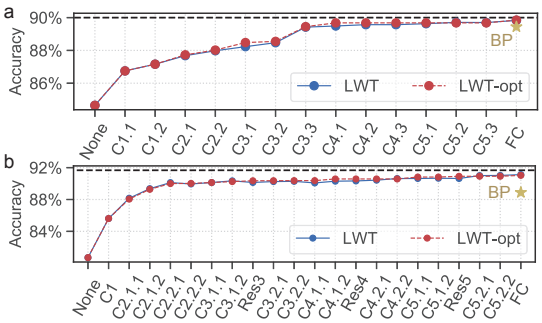

Fig. 4. The training process of the LWT method, the optimized LWT method and compared with the back-propagation algorithm. (a) VGG-16 network (b) Resnet-18 network.

It should be noticed that the recognition accuracy increases faster in the earlier layers. This may come from the varying importance of different layers. To quantify the importance of each layer, we calculate the accuracy difference of different layers in the two perspectives: positive contribution (D_{pc}) and negative deviation (D_{nd}). The D_{pc} of the l^{th} layer is defined as the difference between the accuracy of networks that has trained the l^{th} layer and the initial accuracy of networks with the compact model. The D_{nd} of the l^{th} layer is defined as the difference between the accuracy of networks without non-ideal characteristics and the accuracy of networks with the non-ideal l^{th} layer. As shown in Fig. 5(a), the D_{pc} and D_{nd} of the earlier layers (C1.1 to C3.3) are larger than the back layers (C4.1 to FC). For example, the D_{pc} and D_{nd} of C1.1 are 9.7% and 15.1%, which are around 50× larger than those of C5.3. The large accuracy difference of earlier layers mainly results from the computing error amplification in the inference phase. Besides, ResNet-18 has shown similar results in Fig. 5(b). Comparing Fig. 4 and Fig. 5, it can be found that the accuracy difference of different layers in the neural network determinates the trend of the recovered accuracy curve.

When the accuracy difference of earlier layers is more than that of back layers, a natural idea comes out that the back layers need fewer epochs in the training process. Hence, we proposed the accuracy difference aware LWT method - LWT-opt - to reduce the training epochs without accuracy loss. For example, in the LWT-opt method for ResNet-18, the training epoch of C1-C2.2.2 is heuristically set to 15, and that

of other convolutional layers is 5. As shown in Fig. 4(b), the recognition accuracy of the LWT method and the LWT-opt method is almost the same (91.13% *v.s.* 91.03%). And the average of training epochs in different layers is 6.1 epochs (Fig. 6), which reduces about 70% training time compared to the original LWT method. As Fig. 5 shown, the D_{pc} and the D_{nd} present the same trend of the importance in different layers. Hence, calculating the accuracy difference of D_{nd} before training is an economical method to guide the LWT training process.

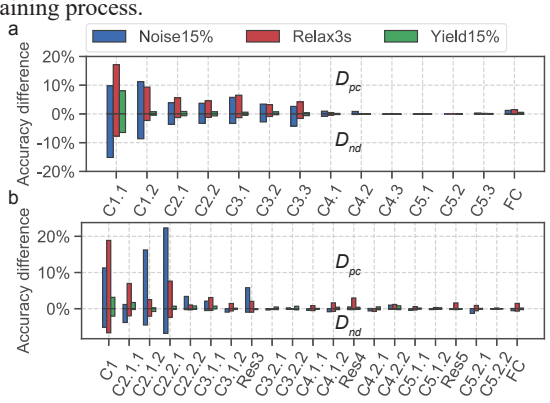

Fig. 5. Accuracy difference of VGG-16 (a) and Resnet-18 (b).

2) Operation and buffer: We evaluate the number of operations and buffers needed in the on-chip back-propagation method and LWT method on different neural networks, including VGG-16 and ResNet-18. In the estimation of operation, it can be divided into inference and training. In the LWT method, the training part consists of the derivation of the activation function, residual, the sign of input feature map, loss function and the inference part need to calculate the convolution of the feature map. Concerning the back-propagation method, the operations of the inference part are the same. But in the training process, except for calculating the derivation of activation function and loss, it needs to calculate the gradient of weight and the residual. As shown in Fig. 7(a), compared to the operations of the back-propagation method, the operations of the LWT are reduced by 75%, 77% in VGG-16 and ResNet-18 networks, respectively. The decrease of operation in the LWT method is mainly contributed by cutting down the residual calculations in the training phase (Blue part in Fig. 7(a)). In Fig. 7(b), the buffer of the LWT method is reduced by 65% and 57%.

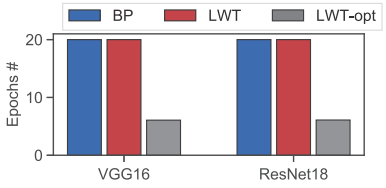

Fig. 6. The training epochs of BP, LWT, and the optimized LWT.

3)Throughput: The proposed pipe-LWT method can improve the throughput of the origin LWT method. To quantify the improvement, we analyze the latency of the origin LWT and the pipe-LWT method in the same task. The latency can be described as followed:

$$T_{LWT} = \left(3Bt + t_{Comp} + t_{Update}\right) \times M \times N, \quad (4)$$

$$T_{pipe-LWT} = (3t + B + t_{Comp} + t_{Update}) \times M \times N, (5)$$

where M is the number of batches in the trainset, N is the number of training epochs. t_{Comp} and t_{Update} are time spent in the updating process. In the LWT and pipe-LWT method, it's natural to train the networks parallelly among different layers. Thus, according to (4) and (5), the pipe-LWT method improves throughput with ~3× speed up than the origin LWT method.

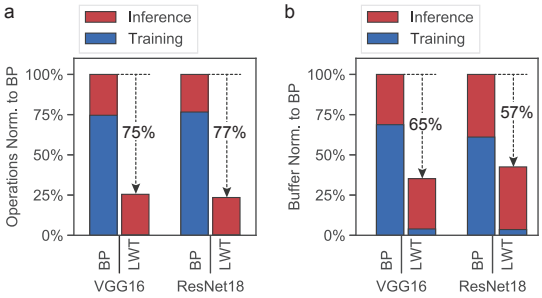

Fig. 7. Operations (a) and buffer (b) of VGG-16 and Resnet-18.

IV. Conclusion and Future Works

In this work, we propose the LWT method, a general on-chip framework with local data to reduce the accuracy loss inducing by the non-ideal characteristics of RRAM device. A compact model for RRAM based on the measured experimental results is proposed. With the decoupling of computing operations, the pipe-LWT method achieves 3× throughput than the origin LWT method. Besides, the accuracy difference-aware LWT method optimizes the training epoch of layers with low accuracy difference. Finally, compared to the back-propagation algorithm, the LWT method can improve the recovered accuracy while the operations and buffer decrease by about 75% and 65%, respectively.

References

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015.

[2] J. Lee et. al., "UNPU: A 50.6TOPS/W unified deep neural network accelerator with 1b-to-16b fully-variable weight bit-precision," in ISSCC, San Francisco, CA, Feb. 2018, pp. 218–220.

[3] W. Zhang et al., "Neuro-inspired computing chips," Nat. Electron., vol. 3, no. 7, pp. 371–382, Jul. 2020.

[4] D. Ielmini and H.-S. P. Wong, "In-memory computing with resistive switching devices," Nat. Electron., vol. 1, no. 6, pp. 333–343, Jun. 2018.

[5] S. Pi et al., "Memristor crossbar arrays with 6-nm half-pitch and 2-nm critical dimension," Nat. Nanotechnol., Nov. 2018.

[6] W. Zhang et al., "Design Guidelines of RRAM based Neural-Processing-Unit: A Joint Device-Circuit-Algorithm Analysis," in DAC, Las Vegas NV USA, Jun. 2019, pp. 1–6

[7] W. Wu et al., "A Methodology to Improve Linearity of Analog RRAM for Neuromorphic Computing," in VLSI Technology, Honolulu, HI, Jun. 2018, pp. 103–104.

[8] C. Li et al., "Analogue signal and image processing with large memristor crossbars," Nat. Electron., vol. 1, no. 1, pp. 52–59, Jan. 2018.

[9] V. Joshi et al., "Accurate deep neural network inference using computational phase-change memory," Nat. Commun., vol. 11, no. 1, p. 2473, May 2020.

[10] P. Yao et al., "Fully hardware-implemented memristor convolutional neural network," Nature, vol. 577, no. 7792, pp. 641–646, Jan. 2020.