Extended Abstracts of the 2019 International Conference on Solid State Devices and Materials, Nagoya, 2019, pp393-394

Impacts and Solutions of Nonvolatile-Memory-Induced Weight Error in the Computing-in-Memory Neural Network System

D. Y. Lee[#], C. H. Wang[#], Y. H. Lin[#], M. L. Wei[*], M. H. Lee[#], H. L. Lung[#], K. Y. Hsieh[#], K. C. Wang[#,*], and C. Y. Lu[#,*]
[#] Emerging Central Lab., [*] Emerging System Lab., Macronix International Co., Ltd.
16 Li-Hsin Rd. Hsinchu Science Park, Hsinchu, Taiwan, R.O.C.
TEL: +886-3-5786688 ext. 78158, FAX: +886-3-5789087, Email: dylee@mxic.com.tw

Abstract

Nonvolatile memories are often used for storing weights in the artificial neural networks (ANN) utilizing computing-in-memory (CIM) architectures. Nevertheless the imperfections of the nonvolatile memory, including the programming error, read fluctuation, and retention loss could induce weight shift and cause significant computing accuracy loss, which then leads to ANN performance degradation. Among various computing operations in the ANN the Batch Normalization (BN) is found to be a good tuning knob to recover inference accuracy under such issues. A simple procedure by calibrating just few BN parameters, without needing to re-adjust the huge amount of shifted weights, can help to restore the neuron distribution and dramatically improve the output accuracy. ReRAM-based convolution neural network (CNN) is used as a vehicle for this study and excellent results are demonstrated.

Introduction

Artificial Intelligence (AI) has drawn many interests recently due to many practical applications enabled by tremendous improvement of computing power and big data. Yet a major bottleneck in the traditional von Neumann computing architecture, the data movement between computing units and memory units, limits the performance and power efficiency of the system [1]. Computing-in-memory architecture is considered as the solution [2] to tackle this problem by performing multiplication and accumulation (MAC) operation directly inside the memory array. However, errors from nonvolatile memories (NVM) such as memory level error (from programming or read fluctuation) and the retention-induced level shift could cause weight value change, leading to system failure [3, 4]. This problem is more significant for inference-only applications [5] because re-training the whole weight array in the field to accommodate the error is very difficult considering the lack of computing power, training time, and even the NVM programming endurance required for training operations. This paper discusses the impacts of memory errors on a ReRAM-based CNN image classifier and demonstrates that the system accuracy can be greatly recovered by simply calibrating the BN parameters.

Experimental Approach

A CNN image classifier with 6 convolution layers and 3 fully connected layers was used to investigate the impacts of weight errors, as shown in **Fig. 1** and **Table 1**. The weight values trained on a server with CIFAR-10 image database training set are converted into conductance level and programmed into the NOR-type CIM ReRAM array (**Fig. 2**)**.** The inference accuracy is evaluated with the test set from the same database. In the inference operation the inputs from previous layer are first translated into voltage levels and applied to the BLs. An ReRAM cell then translates the voltage v into current i based on the Ohmic Law $i = g * v$, where g indicates the conductance of the ReRAM cell. The currents from different memory nodes on the same SL are then collected into the output node which finishes the MAC operation with output current $I = \sum i = \sum g*v$. Three ReRAM error sources are then introduced to the system: program error, read fluctuation, and retention loss. Previous studies [6] suggested that program errors are around 0.07 in standard deviation, and read fluctuations are within 0.01~0.04 in standard deviation, depending on its conductance levels as shown in **Fig. 3**. The amount of retention loss is also suggested in [6]. The server-trained CNN weight values ranging between -0.2 and 0.2 are linearly mapped to the conductance range from 2×10^{-6}S to 4×10^{-5}S, as shown in **Fig. 4**. The activation functions, including BN, Max Pooling, Dropout, and Rectified Linear Units (ReLU), are done by logic computation and the parameters of these functions are store digitally. To simplify the evaluation, the ReRAM error characteristics were introduced into the CNN weight matrix layer by layer.

Results and Discussions

During the inference operation, the test image data are fed into the network without changing weight values. The information is processed and propagates layer by layer through the whole network and the final outcome provides the classification result. The inference accuracy drops from the ideal (floating-point CNN on server) accuracy when ReRAM weight errors are introduced to only one selected layer in the system (**Fig. 5**), especially for the case of the 1st convolution layer where the accuracy drops to an unacceptable value (0.2). In this case the neuron value distribution (i.e. sum-of-product value distribution) strongly shifts to be more negative and broadens significantly, as comparing to the ideal one (from server system) as shown in **Fig. 6**. On the other hand, the accuracy degradation is not as significant if the weight errors are introduced to the 2nd layer or beyond (**Fig. 5**). The results indicate the 1st convolution layer is the most critical layer, thus more prone to weight error, in the CNN system because this layer extracts the local features of the images (whose input values are mostly not zero) and passes the information to the following layers. The errors from the 1st layer would propagate through the rest of CNN layers and it's hard to average-out the error from the network parallelism, as shown in **Fig. 7**. As for the cases other than the 1st layer, the inference accuracy remains high. This is possibly due to the following reasons. ReLU function forces all negative values from previous layer output to zero, which not only discarded the calculation fluctuation from previous layers but also causing weight error in the next layer become invisible. This results in minimum error at output. Another key activation function is the BN which is designed to deal with the covariant shift and scaling after MAC. Experiments showed that BN will enhance error propagation if input errors are not taken into consideration. Conventional training process extracts the mean and variance of the output neuron distribution and stored them as BN parameters (**Fig. 8**). In the inference phase the BN function uses these two parameters to shift and scale the input data to next layer to prevent data explosion. However errors in NVM weights could deviate the layer output while BN parameters from ideal server environment cannot help to recover the distribution shift. An on-demand BN parameter calibration technique (**Fig. 9)** is proposed here to compensate the weight error by updating the mean and variance values of MAC results when the weights are written to the CNN for the first time (i.e. the weight fluctuation), or after long term operation (i.e. the data retention problem). **Fig. 10** shows the significant accuracy improvements of BN calibration when error-bearing ReRAMs are applied to one selected CNN layer, as well as when only performing BN calibration on the first layer for a system with ReRAMs in all the convolution layers. Significant accuracy improvements in the 150°C accelerated retention test are also demonstrated in **Fig. 11**.

Conclusion

The impacts of weight error in CIM system is studied with ReRAM CNN image classifier. The first convolution layer is found most prone to memory level change where the output neuron distribution is shifted and broadened. A BN parameter calibration method is proposed by exploiting the mean and variance of MAC distribution to determine BN parameters. As a result the CNN accuracy can be greatly improved even with errors from NVM level fluctuation and retention shift.

References

[1] J. Hasler, et al., Front. Neurosci., vol.7, art. 118, 2013
[2] X. Guo, et al., IEDM, session 6.5, pp. 151-154, 2017
[3] Y. H. Lin, et al., IEDM, session 2.5, pp. 40-43, 2017
[4] J. Kang, et al., IEDM, session 6.4, pp. 147-150, 2017
[5] A. Mohanty, et al., IEDM, session 6.3, pp. 143-146, 2017
[6] Y. H. Lin, et al., TED, vol. 66 pp. 1289-1295, 2019.
[7] S. Ioffe and C. Szegedy, ICML, volume 37, pp. 448-456, 2015

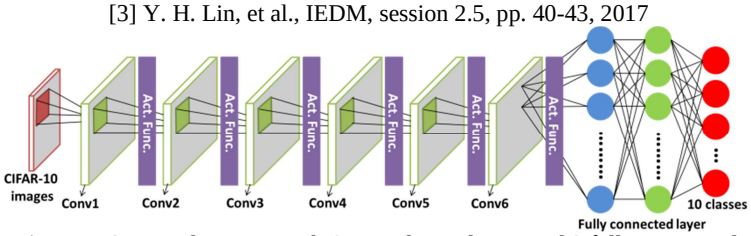


Figure 1 CNN architecture with 6 convolution layers and 3 fully connected layers was used to perform CIFAR-10 image recognition.

	Conv1	Act. Func.	Conv2	Act. Func.	Conv3	Act. Func.	Conv4	Act. Func.	Conv5	Act. Func.	Conv6	Act. Func.	FCL1	Act. Func.	FCL2	Act. Func.	FCL3
Filter/ Neuron	128	BN	128	BN	256	BN	256	BN	512	BN	512	BN	1024	BN	1024	BN	10
		ReLU		Max pooling		ReLU		Max pooling		ReLU		Max pooling		Dropout		Dropout	
				Dropout				Dropout				Dropout		ReLU		ReLU	
				ReLU				ReLU				ReLU					

*BN=Batch Normalization

Table 1 Detailed CNN model including filter numbers for convolution layers, neuron numbers for fully connected layers and activation functions for each layer.

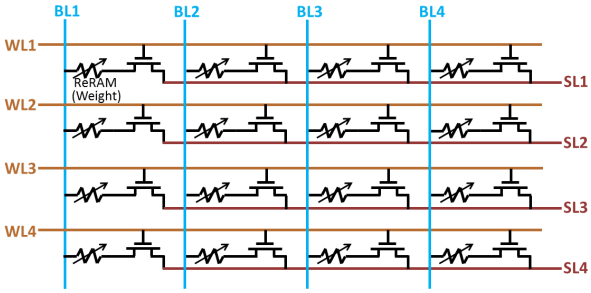


Figure 2 Basic ReRAM array structure where the conductance values of the ReRAM cells represent the analog weights in CNN.

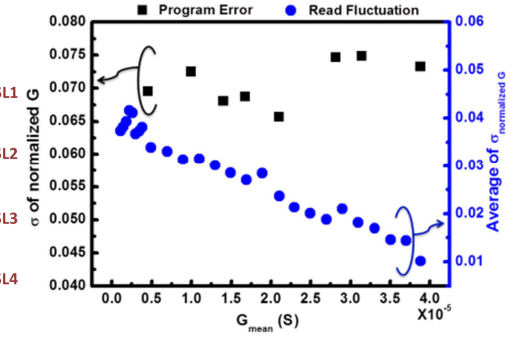


Figure 3 The standard deviation of conductance when programming the ReRAM cell to a specific conductance value (program error) and the conductance fluctuation when reading the ReRAM cell at different conductance states (read fluctuation).

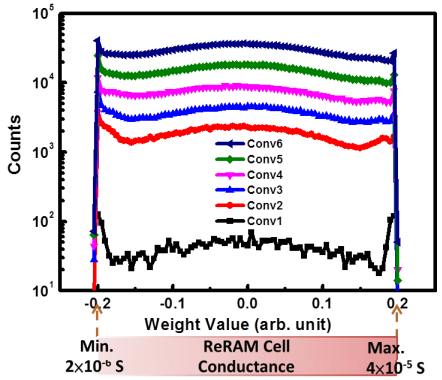


Figure 4 Weight distribution of each convolution layer and conductance range of ReRAM cells linearly map to weight range.

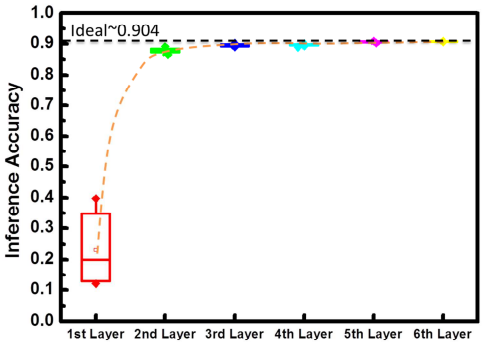


Figure 5 Impacts of program error and read fluctuation in only one assigned convolution layer on inference accuracy. The ideal accuracy using precise floating point weights is around 0.904.

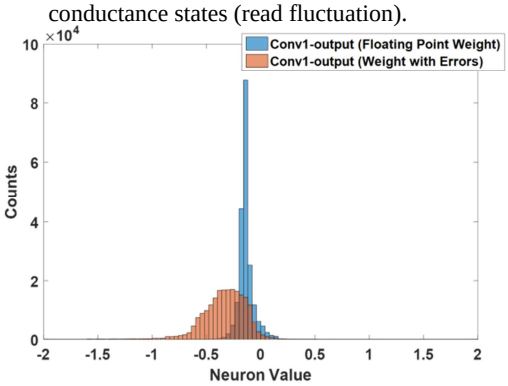


Figure 6 Comparison of the neuron output value distributions, i.e. sum-of-product of ideal weights and weights with errors at 1st convolution layer.

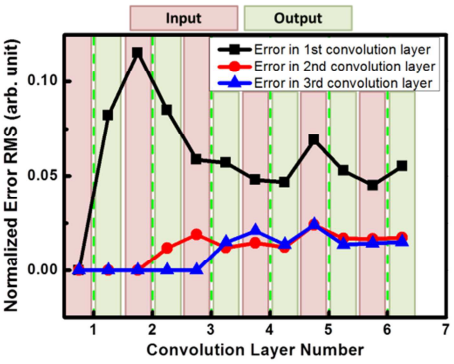


Figure 7 Normalized root-mean-square error in each convolution layer when adding program error and read fluctuation on the 1st, 2nd, and 3rd convolution layer, respectively.

Batch Normalization:

Input: Values of x_i over a mini-batch: B=$\{x_{1...m}\}$; Parameters to be learned: γ, β
Output:$\{y_i=BN_{\gamma,\beta}(x_i)\}$

Original Process:

Mean of mini-batch (without weight errors): $\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i$

Variance of mini-batch (without weight errors): $\sigma_B^2=\frac{1}{m}\sum_{i=1}^{m}(x_i-\mu_B)^2$

Normalize (without weight errors): $\bar{x}_i = \frac{x_i-\mu_B}{\sqrt{\sigma_B^2+\epsilon}}$

Scale and shift (without weight errors): $y_i=BN_{\gamma,\beta}(x_i)=\gamma\bar{x}_i+\beta$

Modified Process:

Mean of mini-batch (with weight errors): $\mu_B' = \frac{1}{m}\sum_{i=1}^{m} x_i'$

Variance of mini-batch (with weight errors): $\sigma_B'^2=\frac{1}{m}\sum_{i=1}^{m}(x_i'-\mu_B')^2$

Normalize (with weight errors): $\bar{x}_i' = \frac{x_i'-\mu_B'}{\sqrt{\sigma_B'^2+\epsilon}}$

Scale and shift (with weight errors): $y_i'=BN_{\gamma,\beta}(x_i')=\gamma\bar{x}_i'+\beta$

Figure 8 Mathematical procedures of batch normalization. The mean and variance of weight errors should be revisited based on the sum-of-product values. [7]

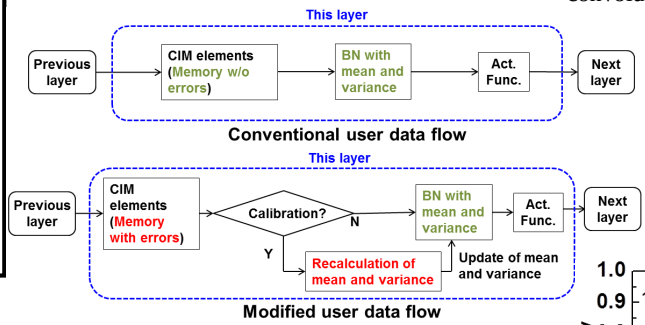
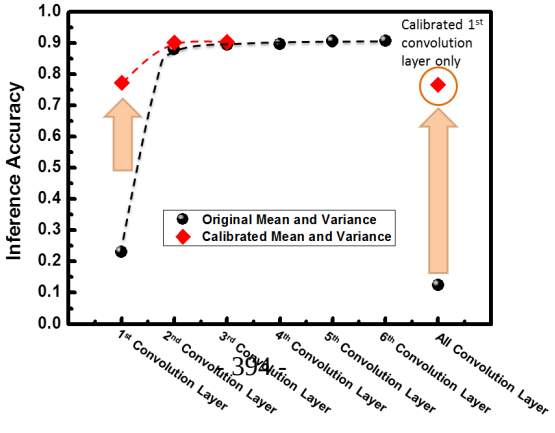


Figure 9 The flow chart of conventional and modified data flow for inference operation. The in-field on-demand calibrated mean and variance are obtained by inputting the training data set to the error-bearing weight matrix, which helps to compensate the weight error.

Figure 10 The inference accuracy before and after modifying the mean and variance parameters. The X-axis label indicates which layer within CNN is replaced with ReRAM. "All convolution layers" mean that ReRAM's are used in all CNN layers. The accuracies are significantly improved with BN calibration.



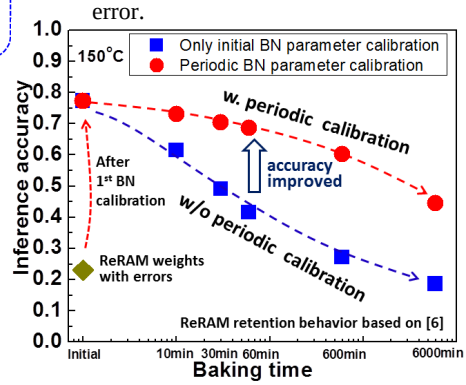


Figure 11 The inference accuracy after baking at 150°C for different time lengths with/without calibrated mean and variance for the 1st convolution layer. The accuracies were significantly improved.