WRAP: Weight RemApping and Processing in RRAM-based Neural Network Accelerators Considering Thermal Effect

Po-Yuan Chen
Dept. of Electrical Engineering
National Cheng Kung University, Tainan, R.O.C.
pychen121@gmail.com

Fang-Yi Gu, Yu-Hong Huang, and Ing-Chao Lin
Dept. of Computer Science and Information Engineering
National Cheng Kung University, Tainan, R.O.C.
{p78101556, p76801360, iclin}@gs.ncku.edu.tw

Abstract— Resistive random-access memory (RRAM) has shown great potential for computing in memory (CIM) to support the requirements of high memory bandwidth and low power in neuromorphic computing systems. However, the accuracy of RRAM-based neural network (NN) accelerators can degrade significantly due to the intrinsic statistical variations of the resistance of RRAM cells, as well as the negative effects of high temperatures. In this paper, we propose a subarray-based thermal-aware weight remapping and processing framework (WRAP) to map the weights of a neural network model into RRAM subarrays. Instead of dealing with each weight individually, this framework maps weights into subarrays and performs subarray-based algorithms to reduce computational complexity while maintaining accuracy under thermal impact. Experimental results demonstrate that using our framework, inference accuracy losses of four DNN models are less than 2% compared to the ideal results and 1% with compensation applied even when the surrounding temperature is around 360K.

I. Introduction

Deep learning has been a hot spot that attracts much attention on artificial intelligence in recent years. Large amounts of data movement between the processing units and the memories are required during the inference of NNs. Hence, large memory access, significant power consumption, and performance degradation are inevitable in the traditional von Neumann architecture. To conquer such a barrier, neuromorphic computing systems (NCS) are often designed with highly parallel and extensively connected computing and storage units to eliminate the gap between CPU computing capacity and memory bandwidth [7].

Computing in memory (CIM) has been considered an energy-efficient approach to implement NCS, which performs data-intensive computations in memory directly and parallelly. By analog computation with the column current summation, one-transistor-one-resistor RRAM crossbar array architecture has O(1) complexity to perform multiplier-accumulation (MAC) computations. Recently, several architectural platforms for RRAM-based NN accelerators have been presented, such as ISAAC [5], PRIME [6]. Although RRAM-based NCS directly inherits many advantages of RRAM, including excellent storage density, low power consumption, excellent scalability, high speed, non-volatility, and high CMOS compatibility, there are unfortunately several issues that need to be resolved, including device defects, variations, and random telegraph noise, and many studies have been done to mitigate these issues [2][3][7]. In addition to these issues, it is necessary to consider the degradation caused by the thermal issues. According to [12], the endurance of the RRAM cell reduces to 0.026x as temperature increases from 300K to 380K and the conductance of RRAM cells degrades significantly as well, which incurs more than 90% inference accuracy degradation at high temperature [10].

Several studies have been devoted to alleviating the temperature-induced negative effects in an RRAM-based NCS [8]-[11]. In this work, aiming at a 3D integration system with offline NN training process, we propose WRAP, a thermal-aware weight remapping and processing framework for RRAM-based NN accelerators to mitigate the heat impact on the RRAM cells while maintaining the inference accuracy of the NN accelerators. The contributions of this work are summarized as follows:

- We propose WRAP, a subarray-based Weight Remapping and Processing framework to alleviate the negative thermal impact on the inference accuracy of RRAM-based NCS accelerators. The proposed framework consists of three algorithms, *weight remapping (WR), weight pruning & splitting (WPS),* and *weight compensation (WC)*.
- WR avoids mapping important weights into higher temperature subarrays. WPS prunes the subarrays with less-critical weights to generate several unused subarrays and then splits the critical weights and maps them into the unused subarrays to protect critical weights.
- To further protects the remaining weights from the thermal effect, we propose WC to map weights into lower conductance level to mitigate the thermal effect.
- We have also validated our proposed framework with several DNN models under different bit-width weight quantization conditions. Experiment results have shown that with the proposed framework, inference accuracy loss of ResNet34 [1], VGG 8, VGG 11, and AlexNet can be less than 2% compared to the ideal results and less than 1% with compensation applied when the surrounding temperature is around 360K.

The rest of this paper is organized as follows. Sec. II introduces the existing RRAM-based NN accelerators and summarizes thermal issues on RRAM cells. The proposed system architecture and simulation procedure are presented in

978-3-9819263-6-1/DATE22/©2022 EDAA

Sec. III and the proposed thermal-aware remapping and processing framework are explained in Sec. IV. Sec. V details the experimental setup and results. Sec. VI concludes this paper.

II. Preliminiaries

A. RRAM-based Neural Network Accelerators

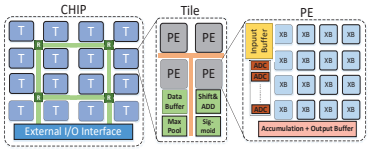
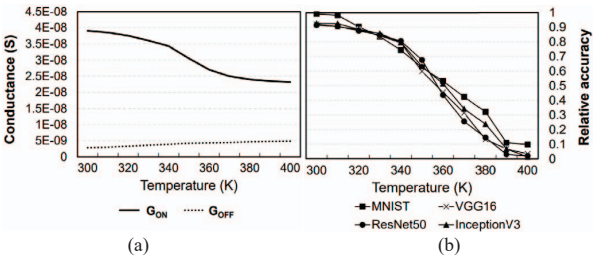


Figure 1. A common system architecture hierarchy for RRAM-based NN accelerators.

Figure 1 shows a common system hierarchy for an RRAM-based NN accelerator. This accelerator usually consists of several tiles (T) connected with routers or a global bus. Each tile has several processing units (PEs) connected with routers or a local bus and some shift-and-adders, sigmoid units, max-pool units, and data buffers (for input/output values). Each PE has several crossbar arrays (XBs, or so-called subarrays), input/output registers, shift-and-adders, and ADCs. High parallelism is one of the advantages of the CIM, hence a hierarchical topology enables high internal bandwidth, reduced data movement, small crossbars (short bit-lines and word-lines to maintain the access speed), and efficient resource partitioning across many layers of a NN for deep learning. Several architectural simulation platforms have been proposed in recent years [5] [6] to perform the architecture exploration for RRAM-based NN accelerators (or NCS).

B. Thermal Impact in RRAM-based Systems



(a) (b)

Figure 2. Temperature impact on (a) RRAM cell conductance and (b) CNN applications relative inference accuracy [10].

Due to the high storage density, low power consumption, and CMOS compatibility, various important RRAM-based applications have been highlighted. However, the negative impact of thermal effects on RRAM is inevitable and should be taken seriously. Walczyk et al. [15] have shown that the ON-state conductance (G_{on}) of an RRAM cell decreases and the OFF-state conductance (G_{off}) increases as the temperature rises. Referring to Figure 2(a), the asymmetry of conductance variation has also been reported: G_{on} drops rapidly while G_{off} increases slowly, and the conductance range starts to drop sharply after 330K–340K. Since the MAC computation depends on the conducting currents of RRAM cells, the asymmetry and significant variations in the cell conductance dramatically degrade the accuracy of an RRAM-based NCS [8][10].

C. Thermal-Aware and Thermal-Resilient Designs

Because of the temperature-sensitive properties of RRAM cells, many studies have focused on thermal-aware solutions and tried to improve the endurance, reliability, lifetime, and inference accuracy of the RRAM-based NN accelerators and NCS [8]-[13]. In [10], the bit-width downgrading (HR^{3}AM-BD) method has been proposed for large-scale DNNs, which discards the LSB stored in RRAM cell at high temperature and shifts current-sum results back in the digital domain. This method degrades the accuracy dramatically at RRAM with low cell resolution since a huge computation error occurs. The tile pairing (HR^{3}AM-TP) method proposed in [10] matches an overheated tile with an idle tile. Both paired tiles are working at low power mode, in which every other row in one crossbar array is activated, such that only half of the cells on a crossbar array are functioning. [10] has enlightened us by showing that protecting the important weights in the crossbar array may be a more efficient and practical way, but practical cell resolution should be kept below 4 or 5 bits [11].

III. Proposed Thermal-Aware Framework

A. System Architecture Scenario

Many studies in RRAM-based NCS architecture exploration have yielded a similar architecture to the Hybrid Memory Cube (HMC) [16] or embedded memory buffers, the processor(s), the RRAM-based NN accelerator(s), and an interconnection network in a 2.5D or 3D stacking integration [6],[9]-[11]. This work uses a similar 3D integration system as shown in Figure 3, which contains an embedded DRAM stack and an RRAM stack side-by-side on top of the host processing unit. The RRAM-based DNN accelerator is located at the bottom layer of the RRAM stack. Given the high operating temperature in the 3D-stacked system, our goal is to develop a weight remapping and processing framework to alleviate the negative thermal impact of RRAM cells, while maintaining inference accuracy.

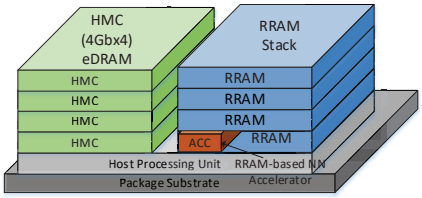


Figure 3. A 3D integration system with an eDRAM stack (HMC-like) and an RRAM stack (including the RRAM-based DNN accelerator in its bottom layer) on the top of a host processing unit.

B. Proposed WRAP Framework

Figure 5 shows the flow of the proposed WRAP framework. We take a DNN model and hardware configuration that includes cell resolution and the size of the RRAM crossbar array as inputs. Then, we obtain the heatmap of the digital layer. After mapping all the weights of the model into the subarrays, we estimate the power of the RRAM cells and the peripheral

circuits. Then, the heatmap is updated with the power of the digital layer and the accelerator. Using the mapping result, the inference accuracy is calculated considering the thermal impact.

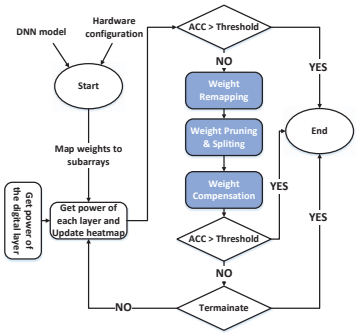


Figure 5. Flow chart of the proposed thermal-aware mapping framework.

If the accuracy, *ACC*, becomes lower than a predefined threshold, the proposed WR will be executed to avoid mapping important weights into higher temperature subarrays. Afterward, the proposed WPS will be executed to prune the subarrays with less-critical weights to generate several unused subarrays. It splits the critical weights and maps them into the unused subarrays to protect critical weights. After WR and WPS, if there still exist unprotected critical weights, WC is conducted for these weights. After the three algorithms are done, the inference accuracy of the DNN accelerator is calculated again. With the new mapping floorplan, if the inference accuracy is still lower than the target threshold, the proposed procedure will be conducted again until the target accuracy is achieved. A termination signal is triggered when the "prune ratio" in the WPS is zero, i.e., the model is not pruned, even if the accuracy is lower than the target threshold. The details of the algorithms are introduced in the following subsections.

C. Weight Remapping (WR)

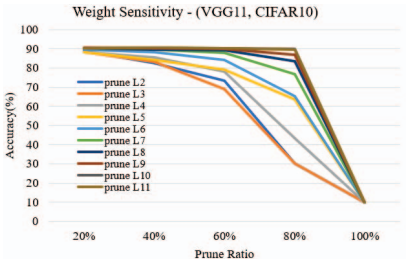


Figure 6. Weight sensitivity of different specific layers with different pruning ratios in the VGG11 model with dataset CIFAR-10.

In the DNN model, the sensitivity of the weights of each layer to the accuracy is quite different. Figure 6 shows the accuracy of the VGG-11 model when different ratios of weights of one specific layer are pruned. The weights of shallow layers (L2, L3, L4) are noticeably more sensitive to accuracy than the deep layers (L8, L9, L10), which shows that weights of the shallower layers have more influence on the accuracy, so we should exert more efforts to protect the weights on the shallower layers, and a straightforward and effective way is mapping these weights on subarrays with lower temperature. For example, in Figure 4, since the subarrays in the bottom right corner have the lowest temperature while those in the top left corner have the highest temperature, weights of each layer, starting from Layer 1 to Layer 4, are mapped sequentially and layer by layer from the subarrays of the bottom right corner to the subarrays of the top left corner. This procedure continues until all weights are mapped into subarrays.

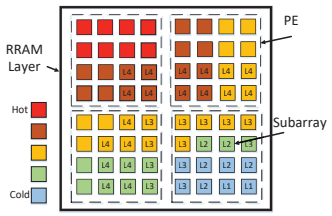


Figure 4. Subarrays are placed layer by layer. Different color means different temperature levels. The subarray with L1 means the weights of Layer 1 are stored in this subarray.

Figure 7 shows the WR algorithm. For each layer, weights are firstly divided into smaller weight sets that can fit in the size of subarrays using the *Divide_weights* function. These sets are stored in *weight_sets*, each of which would be mapped into a subarray. The *Criticality* function estimates the criticality for each weight set, which is estimated by the absolute sum of the weights, and the results are restored in *Criticality_list* (line 3). The larger the sum, the higher the criticality of the weight set.

Because long interconnection distances lead to high power consumption, for practical design consideration, we map the weights of each layer and adjacent layers into subarrays as close as possible to reduce the interconnection delay. We explore the potential mappings starting from each corner of the accelerator in the *Placement* function, and with the heatmap, the most suitable mapping between weights and subarrays can be decided. First, we calculate the number of subarrays required for the weights of the model using the *Cal* function (line 7). For each corner i, we sum up the temperature of subarrays j from corner i and store the summation of the temperature of the subarray into *Corner*[i]. Then the corner i with minimal summation of temperature is stored in *start_corner*. Finally, the weights are mapped into subarrays using the *Placement* function. The mapping result of the i-th corner is stored in a floorplan, *Weight_map*. Finally, the *Weight_map* with the lowest temperature is returned (line 15).

Weight Remapping (WR)
Input: heatmap, DNN_model
Output: Weight_map, Criticality_list

```
1:  foreach Layer i ∈ DNN_model do
2:    Weight_sets[i] = Divide_weight (Layer i)
3:    Criticality_list[i]= Criticality (Weight_sets[i])
4:    Sort the Criticality_list[i]
5:  End
6:  Corner[i] = 0, i= 0~3
7:  Num_sub = Cal(DNN_model)
8:  for each corner i do //analyze heatmap
9:    for j=0 to Num_sub do
10:     Corner[i] = Corner[i] + temperature of subarray j
11:   end
12: end
13: start_corner = argmin_i(Corner[i])
14: Weight_map=Placement(Criticality(Weight_sets[j]), heatmap,
    start_corner)
15: return Weight_map, Criticality_list
```

Figure 7. Proposed thermal-aware weight remapping algorithm.

D. Weight Pruning and Splitting (WPS)

After weight sets are mapped to subarrays, it is possible that the temperature of some subarrays is still high, and the weight

sets in these subarrays still need protection. Hence, we propose weight a pruning and splitting algorithm to further mitigate the thermal impact. The basic idea is to prune less-critical weight sets and generate more space to further protect the critical weight sets If there is sufficient space, we will protect these weights by using two RRAM cells to store a weight, mitigating the thermal impact.

Weight Pruning and Splitting Algorithm (WPS)
Input: Remapped_fp, Critical_list, heatmap, model, Prune_ratio, Dataset, Thr_{Acc}, Acc
Output: Weight_map, Prune_ratio,

```
1:  while Acc < Thr_Acc do
2:      Prune_ratio = Prune_ratio – δ
3:      foreach Layer i ∈ model do
4:          Pruned_model = Prune_subarray(Prune_ratio,
                           Critical_list[i], model)
5:      end
6:      Retrained_model = Retrain(Pruned_model, Dataset)
7:      update Weight_map
8:      foreach Layer i ∈ Retrained_model do
9:        foreach PE ∈ Weight_map do
10:         if there are enough unused subarrays in this PE do
11:           split the weight end
12:         else do
13:           move some weights to unused subarrays in other
              PEs and split the weights. end
14:       end
15:     end
16:     Acc = Update_Acc(Weight_map, heatmap,
              Retrained_model, Dataset)
17: end
18: return Weight_map, Prune_ratio, Acc
```

Figure 8. Proposed thermal-aware weight pruning and splitting algorithm.

To understand which weight set in the same layer affects the accuracy more, we analyze the impact of the criticality of weight sets in the neural network. The criticality of each weight set is calculated by the absolute weight sum of that set and sorted in a list. The top 50% sets in the sorted criticality list are classified as "critical", while the last 50% sets are classified as "less-critical". Our preliminary results show that pruning the weight sets with higher criticality leads to higher accuracy loss. For example, the accuracy drops to around 50% if we prune the top 20% of weight sets in the "critical" group, while the accuracy only drops to 73% if we prune the top 20% of weight sets in the "less-critical" group. Therefore, it is reasonable to prune some less-critical weight sets to protect critical weight sets.

Figure 8 shows the weight pruning and splitting algorithm. Firstly, if inference accuracy is lower than the predefined threshold Thr_{Acc}, δ is subtracted from *Prune_ratio*. If after one round of pruning and splitting (line 2~16), the accuracy has not reached Thr_{Acc}, we will repeat this algorithm by decreasing the *Prune_ratio* until the accuracy is higher than the threshold (line 1). The *Prune_ratio* is initialized to 80% and Thr_{acc} is determined by designers depending on the model. *Prune_subarray* function then prunes some of the less-critical weight sets for each layer to generate unused subarrays. This step decreases the inference power consumption but sacrifices the accuracy. The pruned model is retrained to recover the accuracy by utilizing the inherent fault tolerance of DNN models, and the weight mapping of the retrained model is stored in *Weight_map* (lines 6~7). Then, every PE is checked to see whether there are enough unused subarrays for critical weights to be split or not. If there are, we will protect these critical weights by splitting them into some of those unused subarrays in each PE (lines 10~11). Otherwise, we move these weights in some subarrays from the current PE to an adjacent PE which has sufficient unused subarrays (line 13). Then, a critical weight originally stored in one cell is split into two cells and each cell uses only the lower half conductance levels to represent their weight value, an example will be shown and explained later in this section. Finally, the accuracy after splitting the subarrays is updated (line 16). Figure 9 shows a floorplan example after weight pruning, and Figure 10 displays the floorplan after splitting. The pruned weights and the corresponding subarrays are presented in grey. In Figure 9, the weights in red dashed circles numbered from 1 to 4 cannot be split because there are no sufficient unused subarrays in their PEs. Hence, as depicted in Figure 10, weights assigned to red circles 2 and 4 are moved to and split in the top left PE, and then the weights in red circles 1 and 3 can be split locally.

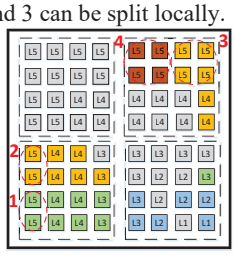


Figure 9. Floorplan after weight pruning. Weights of the subarrays in grey are pruned.

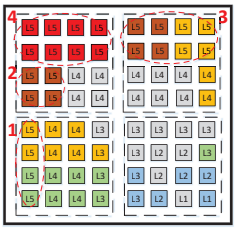


Figure 10. Floorplan after weight splitting of the 5-th layer (L5).

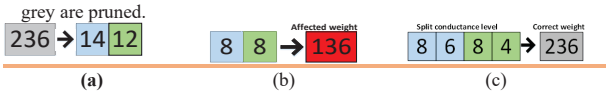


Figure 11. An example of weight splitting. (a) An 8-bit weight is mapped to two 4-bit RRAM cells at 300k. (b) The affected weight is 136 at 400K. (c) The correct weight after WPS at 400K.

Figure 11 shows an example of the weight splitting process. A normalized 8-bit weight, with the decimal value 236, is mapped to two 4-bit RRAM cells. The left and right digits are the MSD and the LSD respectively. Decimal values of the MSD and the LSD are 14 and 12. However, the high conductance values will be decreased to the middle value of the original range at 300K if the temperature rises to 400K. Therefore, the weight value 236 is changed to a wrong value 136 which consists of decimal values 8 and 8 respectively due to the thermal impact, as shown in Figure 11(b). To conquer this problem, we split each digit into 2 RRAM cells by using the lower half of the conductance range only, and the weight split result is shown in Figure 11(c).

E. Weight Compensation (WC)

The prune ratio of some NN layers may be lower than 50%. In this situation, a portion of the weight sets cannot be protected after executing the WPS algorithm, leading to accuracy loss. We propose using WC that is similar to the one in [10], but this

method is only used on the subarray not protected by the previous two algorithms. Since the thermal issue has a higher impact on the higher conductance level of the RRAM cell, weight is shifted right to reduce its value and hence it can be mapped to lower conductance with less thermal effect. After the multiplication results are obtained, the results are shifted left to compensate for the errors caused by the thermal effect.

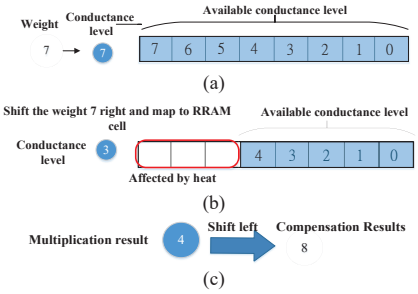


Figure 12. An example of weight compensation.

Figure 12 shows an example of WC. Assume an RRAM cell has 7 conductance levels at a low temperature and weight 7 is mapped to conductance level 7 and the input voltage is 1, as shown in Figure 12(a). As the temperature rises, the three high conductance levels encircled by a red frame are affected by the heat and cannot properly store the weight. Then, the weight 7 is reduced to 4 by shifting right and the value is mapped to conductance level 4. Finally, the multiplication result 4 is obtained, but it is shifted left to obtain the compensated results 8, which is close to the original computed results.

IV. Experimental Results

A. Experimental Setting and Simulation Tools

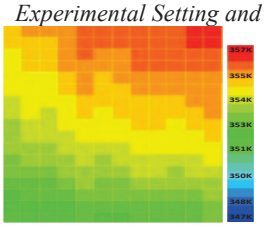


Figure 13. The heatmaps of the RRAM NN accelerator stacked on the digital layer with conductance values from [8]

For the thermal analysis under our system assumption, we refer to the area and power parameters of the Intel Xeon E5-2670-v3 CPU (operating at 1.2GHz and consuming 120 watts) as the host processing unit, and that of the Micron 2GB DDR4 DRAM HMC [16] as the 4-layer HMC-like eDRAM stack, and that of the hierarchical architecture in ISAAC [5] with RRAM crossbar arrays of size 128 by 128 as our RRAM-based DNN accelerator. Parameters of the peripheral circuits and RRAM subarrays in the RRAM-based DNN accelerator are referred from [5] and [17] with results operating at 1.2GHz. We use the validated thermal simulator, Hotspot 6.0 [18], to perform the thermal analysis with the floorplan and power information mentioned above, and the ambient temperature is set to 318K. We simulate heatmaps of our system for RRAM cells with conductance ranges [3.07nS, 38.4nS][8] as shown in Figure 13. This is a common range used in several previous works mentioned in Sec. II. To validate the proposed thermal-aware algorithms, we build an RRAM-based NN accelerator simulator in Python, which is based on the Pytorch framework [19]. We evaluate our accelerator on 4 DNN models with large-scale neural networks (VGG8, VGG11, ResNet34[1], and AlexNet) for CIFAR-10 classification.

B. Experimental Results

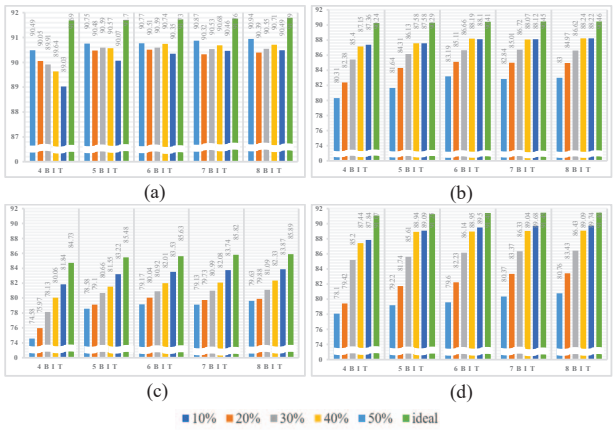


Figure 14. Inference accuracy under the thermal impact of (a) VGG8, (b) VGG11, (c) AlexNet, and (d) Resnet34 after using proposed thermal-aware WR and WPS algorithms with different cell resolutions.

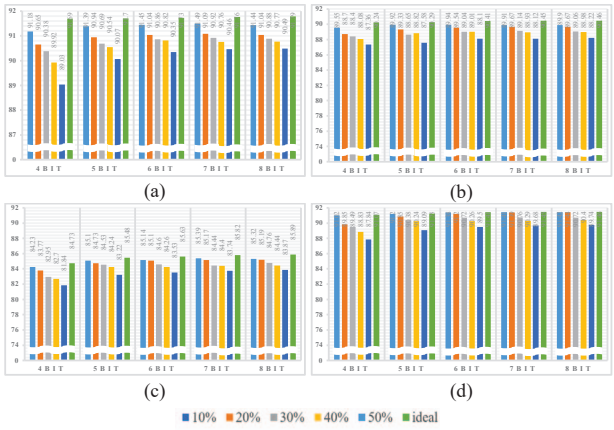


Figure 15. Inference accuracy under the thermal impact of (a) VGG8, (b) VGG11, (c) AlexNet, and (d) Resnet34 after using compensation with different RRAM cell resolutions.

The weight floorplans with the highest accuracy after the WR algorithm are used to evaluate the proposed WPS algorithm. In Figure 14, with 4 to 8 bits cell resolution, the proposed WR and WPS algorithms can minimize the accuracy loss, compared to the ideal case, to less than 2% for all models with different pruning ratios. After the WPS algorithm, models pruned with a low pruning ratio result in more weights not being protected. Therefore, the accuracy of the low pruning ratio is slightly lower than the high pruning ratio. The best results are listed in this form *(model, prune ratio, cell resolution, ideal accuracy, accuracy loss)* for the four models as follows: *(VGG11, 40%, 8bits, 88.44%, 2%), (VGG8, 40%, 8bits, 90.79%, 1.00%), (Alexnet, 40%, 4bits, 81.84%, 1.78%), and (ResNet34, 40%, 6bits, 89.50%, 1.77%)*. The best results after WR and WPS happened when the pruning ratio is around 40% to 50%. Even with the limitation of interconnection distance, the proposed thermal-aware WR and WPS algorithms are suitable for a wide

cell resolution range and could efficiently protect the RRAM cells from the thermal impact for different DNN models.

The accuracies of most models decline when the prune ratio is less than 50%. Those unprotected weights suffer from thermal issues and decrease the accuracy. Hence, we compensate these weights using WC. Figure 15 shows that with the WC executed, the accuracies resume in four NN models with different quantization bits widths, and all of them succeed in less than 1% accuracy loss.

C. Comparisons

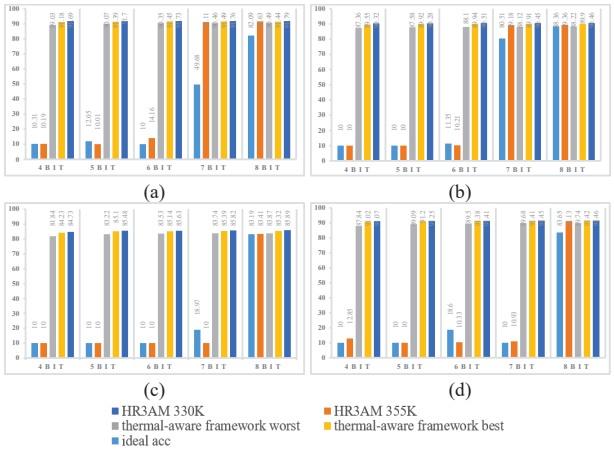


Figure 16. Comparison of the proposed thermal-aware methodology and HR3AM with different cell resolutions and the inference accuracy under the thermal impact of (a) VGG8, (b) VGG11, (c) AlexNet,(d) ResNet34.

In this section, we compare our algorithms to HR3AM [10]. In Figure 16, the best and the worst results of the proposed framework with different cell resolutions, which are referred from Figure 14. The ideal result of each case is depicted in light blue. Because the surrounding temperature could be up to 350K to 360K in our system, we simulate the results of HR^{3}AM with two different active thresholds 330K (in dark blue) and 355K (in ocher red). It is apparent that the accuracy loss of HR^{3}AM using both thresholds is significant with low cell resolution because much information is lost while discarding the LSB of weights in overheated subarrays, whose temperature is higher than the active threshold. HR^{3}AM is more suitable for DNN with high cell resolution. The comparison results in Figure 16 show that the proposed framework allows for lower accuracy loss than HR^{3}AM and is suitable for a variety of situations.

V. Conclusions and Future Work

Emerging RRAM is one of the most promising memories for CIM applications. However, the thermal impact of the RRAM degrades the conductance range and should be taken seriously. In this work, we proposed WRAP, a subarray-based thermal-aware weight mapping algorithm that is efficient and practical for the 3D integration system. The proposed framework consists of thermal-aware weight remapping (WR), weight pruning & splitting (WPS), and weight compensation (WC). The proposed method is verified with several DNN models under different cell resolutions. Experimental results showed that the proposed framework achieves less than 2% inference accuracy loss, and less than 1% loss if applied compensation for VGG 8, VGG11, Alexnet, and ResNet34. Although only temperature-induced conductance degradation is considered, our future work will consider other reliability and variation issues and propose methods to tackle these issues in our framework.

Acknowledgement

This work is supported in part by the Ministry of Science and Technology under grant MOST 109-2628-E-006-012-MY3 and 110-2221-E-006-084-MY3 and in part supported by the Intelligent Manufacturing Research Center (iMRC) from the Featured Areas Research Center Program by the Ministry of Education, Taiwan (ROC).

References

[1] K. He et al., "Deep residual learning for image recognition," *in* 2016 CVPR, pp. 770–778.

[2] J. Lin et al., "Rescuing RRAM-based Computing from Static and Dynamic Faults," in IEEE Trans. CAD, early access, oi: .1109/TCAD.2020.3037316.

[3] C. Liu et al, "Rescuing memristor-based neuromorphic design with high defects," in 2017 DAC, pages 1-6

[4] X. Peng, R. Liu, and S. Yu, "Optimizing Weight Mapping and Data Flow for Convolutional Neural Networks on RRAM Based Processing-In-Memory Architecture," in 2019 ISCAS, pp. 1-5

[5] A. Shafiee et al., "ISAAC: A Convolutional Neural Network Accelerator with In-Situ Analog Arithmetic in Crossbars," in 2016 ISCA, pp. 14-26.

[6] P. Chi et al., "PRIME: A Novel Processing-in-Memory Architecture for Neural Network Computation in ReRAM-Based Main Memory," in 2016 ISCA, pp. 27-39.

[7] F. Zahoor, T.Z. Azni Zulkifli and F.A. Khanday, "Resistive Random Access Memory (RRAM): an Overview of Materials, Switching Mechanism, Performance, Multilevel Cell (MLC) Storage, Modeling, and Applications," *Nanoscale Res Lett 15*, 90, 2020.

[8] M. V. Beigi et al., "Thermal-aware optimizations of ReRRAM-based neuromorphic computing systems," *in 2018 DAC*, pp. 1–6.

[9] M. Zhou, M. Imani, S. Gupta, and T. Rosing, "Thermal-Aware Design and Management for Search-based In-Memory Acceleration," *in 2019 DAC*, no. 174, pp. 1–6.

[10] Xiao Liu et al., "HR^{3}AM: A Heat Resilient Design for RRAM-based Neuromorphic Computing," in 2019 ISLPED, pp. 1-6

[11] H. Shin, M. Kang, and L.-S. Kim, "A thermal-aware optimization framework for ReRAM-based deep neural network acceleration," in 2020 ICCAD, pp. 1-9.

[12] M. V. Beigi and G. Memik, "THOR: THermal-aware Optimizations for extending ReRAM lifetime," in 2018 IPDPS, pp. 670–679.

[13] S. Zhang, G. L. Zhang, B. Li, H. Li, and U. Schlichtmann, "Lifetime Enhancement for RRAM-based Computing-In-Memory Engine Considering Aging and Thermal Effects," in 2020 AICAS, pp. 11-15

[14] P.-Y. Chen, X. Peng, S. Yu, "NeuroSim+: An integrated device-to-algorithm framework for benchmarking synaptic devices and array architectures," in 2017 IEDM, pp.6.1.1-6.1.4.

[15] C. Walczyk et al., "Impact of Temperature on the Resistive Switching Behavior of Embedded HfO2-Based RRAM Devices," *IEEE Trans. Electron Devices*, vol. 58, no. 9, 2011.

[16] Hybrid Memory Cube (HMC) Gen2, Micron Technology Inc. https://www.micron.com/-/media/client/global/documents/products/data-sheet/hmc/gen2/hmc_gen2.pdf

[17] C.-X. Xue et al.,"A 1Mb Multibit ReRAM Computing-In-Memory Macro with 14.6ns Parallel MAC Computing time for CNN-based AI Edge Processors," 2019 ISSCC, Dig. Tech. Papers, pp. 388-390

[18] R. Zhang et al., HotSpot 6.0: Validation, Acceleration and Extension," *University of Virginia, Tech. Report*, CS-2015-04.

[19] A. Paszke et al., "PyTorch: An Imperative Style, High-Performance Deep Learning Library," Advances in Neural Information Processing Systems 32, 2019, pp.8024– 8035.