HR^{3}AM: A Heat Resilient Design for RRAM-based Neuromorphic Computing

Xiao Liu Mingxuan Zhou Tajana S. Rosing Jishen Zhao
Department of Computer Science and Engineering, University of California, San Diego
{x1liu, miz087, tajana, jzhao}@ucsd.edu

***Abstract*—RRAM based accelerators have been widely adopted in many neuromorphic designs. However, RRAM cells are sensitive to temperature, which changes RRAM's conductance. Such heat-induced interference can significantly decrease the computational accuracy because values are functions of RRAM conductance. In this paper, we propose HR^{3}AM, a heat resilience design, which improves accuracy and optimizes the thermal distribution of RRAM based neural network accelerators. HR^{3}AM consists of two key mechanisms: *bitwidth downgrading* and *tile pairing*. Bitwidth downgrading re-represents weights by shifting the conductance to improve the network inference accuracy. Tile pairing matches hot crossbar units with pre-defined idle units to mitigate high-temperature issues. We evaluated HR^{3}AM on four real world neural network models. Results show that HR^{3}AM improves classification accuracy by up to 41.8% compared with current state-of-the-art designs. For thermal optimization, HR^{3}AM effectively decreases the maximum temperature by 6.2K and average temperature by 6K.**

I. Introduction

Neuromorphic computing has attracted increasing attentions today because the applicable scenarios of the neural network continue to expand. Convolutional neural networks (CNNs) are widely adopted because of their high prediction accuracy. As network sizes grow, general-purpose hardware platforms, such as CPU and GPU, are insufficient for delivering decent performance and power-efficiency for neuromorphic computing applications. There have been a wide variety of neuromorphic computing accelerators, which utilize fast and power-efficient emerging technology to accelerate large-scale CNNs.

Resistive RAM (RRAM), or memristor, have been widely investigated to serve as a CNN accelerator [16]. RRAM can form a crossbar array to perform matrix multiplication by exploiting analog characteristics of RRAM cells. Matrix multiplications are the majority of operations in convolution. Optimizing matrix multiplications is essential to improving the performance of CNN. In a RRAM based CNN accelerator, weights of the matrix are programmed into the conductance of each RRAM cell in each crossbar. The input data is converted to analog signals and transferred to crossbar bitlines. The output data is generated and converted to digital signals on wordlines. Matrix multiplication achieves higher speed and parallelism on RRAM crossbars than conventional hardware [8]. RRAM crossbars also have cost and energy efficiency advantages [16].

However, the thermal issue of RRAM crossbars potentially reduces the inference accuracy of neural networks. Prior work discovered RRAM conductance is sensitive to temperature [18]. RRAM ON state conductance (G_{ON}) and OFF state conductance (G_{OFF}) varies as the temperature changes. The conductance range $[G_{OFF}, G_{ON}]$ sharply declines on the hot cells. This variation drastically decreases weight precision when it is represented in the form of conductance. When the chip continuously operates, the accumulated heat affects more RRAM cells. This causes many weights to be misrepresented during inference. Eventually, neural networks suffer from loss of accuracy. Our experiments show that the accuracy loss at high temperature can be as high as 90%.

978-1-7281-2954-9/19/$31.00 ©2019 IEEE

A few research paid attention to the problem. Most existing works focus on handling device defects and variations in RRAM crossbars [5], [12], [13]. However, because these schemes are based on permanent errors from process variation, they do not dynamically adjust to the defected cells. Temperature aware row adjustment (TARA) [4] proposed a heat resilient design but suffers from several major drawbacks. TARA still needs to be aware of the exact weights and adjusts row order based on each weight's magnitude, which is not scalable with large scale networks. Besides, TARA provides no optimization over thermal issue. As network size expands, the thermal issue gets worse and row adjustment becomes less effective. In this paper, we introduce a novel heat resilient design HR^{3}AM for the RRAM crossbar. HR^{3}AM solves the issue from two aspects: adapting weights to the reduced conductance and optimizing chip thermal distribution. Therefore we correspondingly propose two mechanisms in HR^{3}AM from these two aspects. The first mechanism *bitwidth downgrading* aims to improve accuracy. It adapts weights to the decreased conductance range by reducing bitwidth per RRAM cell. The second mechanism *tile pairing* aims to tackle the thermal issue. It optimizes the chip thermal status by matching hot spot tiles with idle tiles. The contributions of this paper include:

- We evaluate the impact of heat on the inference accuracy of several large scale neural networks. Our results show the inference accuracy of several distinguished CNNs drop to less than 10% of theoretical accuracy.
- We propose HR^{3}AM, which contains two mechanisms: *bitwidth downgrading* and *tile pairing* to improve inference accuracy and tackle the thermal issue, respectively. Bitwidth downgrading increases the accuracy of weights represented in conductance. Tile pairing releases the heat on hot spot tiles.
- We evaluated HR^{3}AM with four state-of-the-art CNN models. The results show our design guarantees 87.8% of the theoretical inference accuracy. Our design also shows a 6.2K decrease on maximum temperature and a 6K decrease on average temperature for the entire chip.

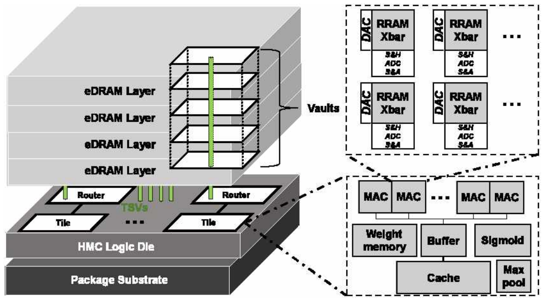

Fig. 1. The structure of HMC RRAM.

II. RRAM CNN Accelerator

A. Architectural Overview

This work explores the RRAM CNN accelerator based on Micro's Hybrid Memory Cube (HMC) [10], [16]. As Figure 1 shows, the RRAM HMC structure contains multiple layers of embedded DRAM (eDRAM) and a single logical layer at bottom. The logic layer contains multiple tiles which are basic computation units for handling CNN operations. Each tile consists of multiple multiply accumulators (MACs), each of which contains multiple RRAM crossbars for matrix multiplications. Tiles also include memory for synaptic weights, caches and buffers for input and output data, a max pool unit and a sigmoid unit [16]. The logic layer accesses DRAM layers with Through-Silicon-Vias (TSVs). Each eDRAM layer is divided into multiple vaults; each vault can independently process memory accesses. Each vault has a router on the logic die for data transitions between eDRAM and tiles.

In each MAC, RRAMs form a crossbar structure. As Figure 5 shows, a memristor connects with a bitline and a wordline. Each memristor has a conductance $g_{i,j}$, where i represents the bitline index and j represents the wordline index. According to Kirchoff's Law, the crossbar is able to perform $V_O = V_I^T * G * R_s$, where G is the conductance matrix, V_O and V_I are output/input voltage respectively. To compute matrix multiplications in CNN, the weight matrix is programmed into memristors and represents G. Input data is converted by digital to an analog convertor (DAC) and feed into a crossbar as V_I. Lastly, the wordline currents are held by a sample-and-hold (S&H) circuit, they are fed to ADC and the output is in the form of V_O.

B. Neural Network Data Mapping on RRAM crossbar

The neural network consists of various types of layers. Most of layers in a deep neural networks are convolutional layers (e.g. 13 out of 16 in VGG16), which conduct a large number of matrix multiplications. Utilizing RRAM crossbar through matrix multiplications in the neural network can increase parallelism and decrease latency. Encoding a weight value into the RRAM crossbar requires conversion between the value and the cell conductance, which follows the formula [8], where $\alpha = \frac{G_{max} - G_{min}}{w_{max} - w_{min}}$ and $\beta = G_{max} - \alpha * w_{max}$:

$$G = \alpha * W + \beta \tag{1}$$

α linearly scales weights to match range of conductance and β adds the offset to remove negative values in weights. The precision of weights is limited by the bitwidth of RRAM cells. Single cell only achieves at most seven bits accuracy [3]. To achieve higher precision, each weight can be programmed into multiple RRAM cells [16].

Each layer of the neural network is serialized. Therefore we can only perform parallel operations within the single neural network layer. Each layer of the neural network should be mapped to a group of RRAM tiles [16]. Because tiles belonging to the different layers are independent, layers can form a pipeline to achieve better throughput [16]. The pipelined RRAM crossbar chip processes multiple inputs concurrently with different layers.

C. Thermal Issues in RRAM Accelerator

RRAM cells originally obtain ON and OFF states. Each RRAM cell can represent multiple bits by setting an intermediate state that has the conductance between ON and OFF states [8]. Each status has its unique conductance range that does not overlap with others. However, temperature changes has a significant impact on the RRAM conductance between OFF and ON states [10]. Increased temperature leads to a narrower range of conductance.

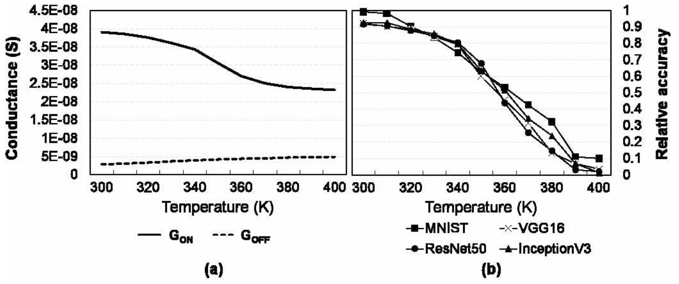

Fig. 2. Temperature impact on (a) RRAM cell conductance and (b) CNN applications relative inference accuracy.

Figure 2(a) depicts this effect on OFF and ON conductance (G_{OFF} and G_{ON}). The ON state has a weak metallic-like characteristic [10] such that G_{ON} significantly drops as the temperature rises. While G_{OFF} increases with temperature because of increased current [10]. The conductance range $[G_{OFF}, G_{ON}]$ drops by 50% when the temperature rises from 300K to 400K. The conductance range starts to drop sharply after 330K, which is a common operational temperature for many chips. We observe the asymmetry of conductance variation: G_{ON} drops rapidly while the G_{OFF} increases slowly.

The shifting conductance leads to weight misrepresentation of the neural network. The weight conversion (Equation 1) between weights and RRAM conductance is based on ideal condition at room temperature. When the temperature changes, certain weight values fall into unavailable conductance range, the weight conversion maps these weight values to the nearest available conductance. These weight values are misrepresented and share conductance with others. We modeled the effect and tested on several large scale neural networks [7], [17]. Figure 2(b) shows the relative reference accuracy of four neural networks under 300K to 400K. To get relative accuracy, we normalize the ReRAM generated accuracy to the software generated accuracy. We assume entire chip is effected by the same temperature. The accuracy drops as the temperature raises. Despite minor variance between different network models, the inference accuracy of all experiments eventually drop below 10% at 400K.

When pipelining multiple CNN applications, each application runs in different layers during runtime. Because different CNN layers have various sizes, the amount of data and matrix multiplication operations conducted by different neural

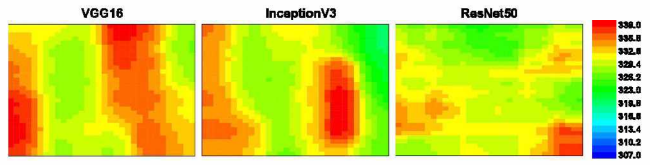

Fig. 3. Temperature distribution of a single RRAM chip when running VGG16, InceptionV3 and ResNet50.

network layers varies. When each layer is bonded to certain groups of tiles on the chip, power consumption varies between tiles and dynamically change with time, which may cause a non-uniform thermal distribution on RRAM chip. We generate the steady state temperature distributions of entire RRAM chip with three CNN models (VGG16, InceptionV3, ResNet50) conducting inference for 10000 ImageNet [15] figures. As Figure 3 shows, the temperature difference can be as high as 17.16K. The distribution also shows several distinctive hot spots. Design-time mechanisms are not sufficient to solve such dynamic issues. Furthermore, identifying hot spot tiles and mitigating heat on hot spot tiles could optimize the thermal status. In the next section, we propose two mechanisms, one can dynamically mitigate the negative impact of overheating, and the other can reduce hotspots on the RRAM chip.

III. HR^{3}AM Design

A. *HR^{3}AM Overview*

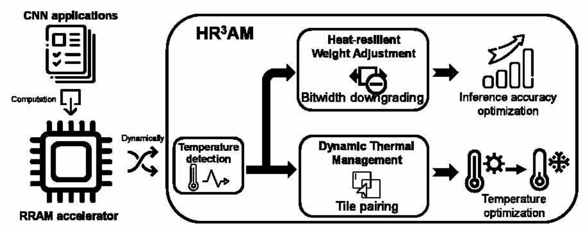

Fig. 4. Overview of HR^{3}AM structure.

To handle thermal issue imposed inference accuracy loss, we propose HR^{3}AM, a heat resilient design for RRAM based CNN accelerators. Figure 4 shows an overview of HR^{3}AM. When running CNN applications, HR^{3}AM monitors the dynamic thermal distribution of the RRAM chip. We adopt the temperature sensor design in commercial processors [14], which provides temperature detection of each crossbar unit. HR^{3}AM provides two heat resilient methods, heat-resilient weight adjustment and dynamic thermal management, to optimize inference accuracy and temperature distribution respectively. For heat-resilient weight adjustment, we introduce a mechanism, called *bitwidth downgrading*, to dynamically change the bitwidth of RRAM cells. Downgrading bitwidth of hot RRAM cells would trade the precision of influenced weights for mis-representation of weights caused by conductance range reduction. For dynamic thermal management, we propose the *tile pairing* to move a portion of operations in hot tiles to pre-allocated idle tiles. Hence, the power consumption of hot tiles would be reduced to release heat. The following of this section discusses details of these two methods.

B. *Heat-resilient Weight Adjustment*

Bitwidth Downgrading. As we discussed in Section II-C, the major cause of accuracy decline is the conductance range drop when temperature rises. Our evaluation shows CNN applications have 0.9% inference accuracy loss on average for every 1K increase on temperature. To accommodate the shifting conductance range, we modify β and α of Equation 1. Based on the observation that G_{OFF} changes much less than G_{ON}, we can get $G_{OFF}^{new} \approx G_{OFF}^{old}$. When temperature rises, the new formula can be calculated with a new coefficient γ:

$$G_{new} = \gamma * (\alpha * W + \beta) \quad (2)$$

where $\gamma \approx \frac{G_{ON}^{new}}{G_{ON}^{old}}$. However, the RRAM conductance range, which is divided into multiple levels to represent all the bits states, are non-adjustable. The conductance shrinkage causes the weight value mapped to a high conductance to be mis-represented as a lower conductance. Instead of directly using these misrepresented values, we propose to downgrade the bitwidth such that converted weight can be properly mapped into available conductance range. Bitwidth downgrading acts as γ in Equation 2. When both weight W and parameter β shift right N bits, the mechanism changes the conversion equation to: $G_{new} = 1/2^N * (\alpha * W + \beta)$, where N represents the number of shifted bits.

Our design also accommodates the multiplication result produced by adjusting weights to ensure output correctness. We directly shift N bits back on the result using existing shift-and-add units. The new formula for output voltage when the bitwidth downgrading effects is as follow: $V_O = V_I^T * G_{new} * R_S * 2^N = (V_I^T * R_S * G_{old}/2^N) * 2^N$. The disadvantage of this mechanism is the loss of weight accuracy. However, bitwidth downgrading uniformly and linearly changes weights. Therefore the ratios between different synapses' weights are unchanged such that calculations using bitwidth downgrading provide better accuracy than those using misrepresented values. As we show in Section IV, bitwidth downgrading causes much less inference accuracy loss than that of temperature induced conductance changes.

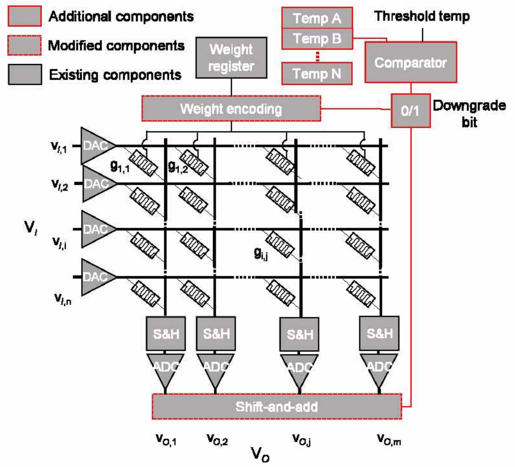

Fig. 5. Bitwidth downgrading hardware in the crossbar array.

Hardware Support. Figure 5 presents the hardware design for bitwidth downgrading. We add a temperature register, a comparator, a *downgrade bit* for each crossbar array and a control circuit. The temperature register stores the most recent sensed temperature of the crossbar array. The comparator checks if the current temperature of the crossbar array exceeds the threshold. The control signal from comparator updates the downgrade bit to notify the crossbar to conduct bitwdith downgrading. The weight encoding hardware reprograms weights to the crossbar cells. We slightly modify weight encoding logic such that the weights shift N bits left before encoding when the downgrade bit is set. Therefore the encoded conductance is $\frac{1}{2^N}$ of the original. The adjustment on the output only requires minor modification on the shift-and-add unit. The control signal from the downgrade bit forces the shift-and-add unit to shift N bits right, such that the final result expands 2^N times and matches with the output of no bitwidth downgrading. The implementation cost consists of a temperature register, a comparator, a downgrade bit and its control logic, and modification to the encoding logic to support bit shifting for each crossbar array. These are either small size storage or simple logic circuits.

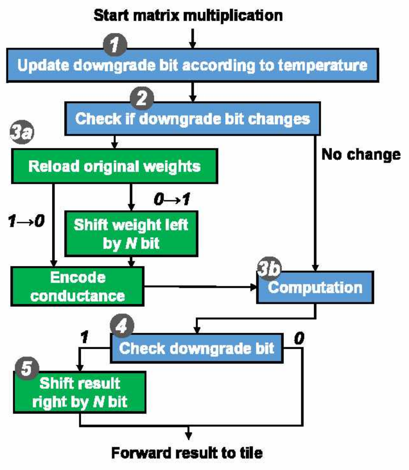
Start matrix multiplication
1
Update downgrade bit according to temperature
2
Check if downgrade bit changes
3a
Reload original weights
No change
0→1
1→0
Shift weight left by N bit
3b
Encode conductance
Computation
4
1
Check downgrade bit
0
5
Shift result right by N bit
Forward result to tile

Fig. 6. Bitwidth downgrading operation flowchart.

Bitwidth downgrading procedure. Figure 6 presents the flowchart of the bitwidth downgrading mechanism on a crossbar array. At step 1, the comparator updates the downgrade bit based on the current temperature. Step 2 checks if the status of the downgrade bit changes, if it is unchanged the crossbar array directly starts multiplication by going to step 3a, while a true state leads to step 3b (bitwidth downgrading/restoration). There are two cases in step 3b. When the downgrade bit makes $0 \rightarrow 1$ change, the crossbar array recomputes conductance with N bit shifted weights and encode them to RRAM cells. When the bit makes $1 \rightarrow 0$ change, the crossbar array conducts weight restoration: it loads original weights and encodes them to RRAM cells. Step 3a conducts matrix multiplication within the crossbar. Step 4 checks the downgrade bit. A true state leads to step 5, the crossbar array shifts N bits back on the output using the shift-and-add unit. The final result is forwarded to tile buffer.

The bitwidth downgrading mechanism only effects inside the crossbar array. It requires neither the coordination between the crossbar array and the MAC, nor the coordination between bitwidth downgraded crossbar arrays. The downgraded weights are generated temporarily and avoid overwriting the original copy. We only shift one bit to match the shifted conductance in the worst case (400K) as we observed in Section II-C. Therefore, we set N=1 across the entire design. We choose the temperature threshold of 330K because we observe a significantly increased accuracy decline rate after 330K on Figure 2.

C. Dynamic Thermal Management

Tile Pairing. As we demonstrate in Section II-C, thermal distribution is not uniform on the RRAM chip. Temporally reducing power consumption on the hot spot tiles may improve the thermal status. However, conventional power saving techniques such as decreasing frequency requires complicated design and significantly decreases performance. Therefore, we propose a novel approach — tile pairing to match an overheated tile with an idle tile, while both paired tiles are working at low power mode. In the low power mode, every other row in one crossbar array is activated, such that only half of the cells on a crossbar array are functioning. Based on [16], a low power mode tile only consumes 61.8% of the power of a full power mode tile. We pair overheated units at the tile level. Because pairing at a finer granularity (MAC or crossbar array) leads to overhead for tracking paired units.

We identify an overheated tile and a cooled-down tile with collected temperature statistics. When 80% of the crossbar arrays within an unpaired tile reach the threshold, we identify the tile as overheated. When the percentage of the overheated crossbar arrays in a paired tile drops below 40%, we identify the tile as cooled-down. A cooled-down tile unpairs with its paired tile. The paired tile resumes as an idle tile.

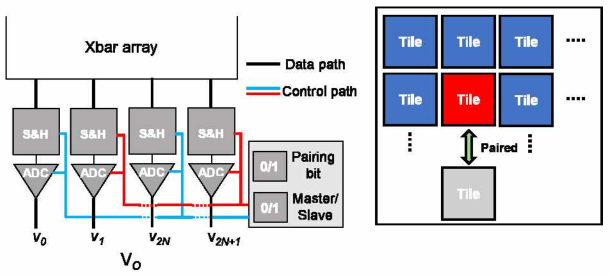

Fig. 7. Tile pairing (a) mechanism and (b) control logic.

Figure 7(a) demonstrates the paring mechanism. When an overheated tile pairs with an idle tile, the hot tile is the master tile and the idle tile is the slave tile. Every crossbar array in the master tile pairs with a crossbar array with same index in the slave tile. Both master and slave tiles use half of their cells to produce the result. Two crossbar arrays use the same weights G and same input V_I. The master tile array uses even-index columns and the master tile array uses odd-index columns. The master tile array produces output $V_O^m = \{v_0, v_2 ... v_{2N}\}$ and the slave tile array produces output $V_O^s = \{v_1, v_3 ... v_{2N+1}\}$. The final result is the union of the two outputs:$V_O = V_O^m \cap V_O^s$, which is constant with result generated from an unpaired crossbar.

To scatter a matrix multiplication to two tiles, we rearrange the data placement between master and slave tiles. During pairing, the controller transfers the weight matrix from weight memory of the master tile to the slave tile's. During multiplication, the master and the slave tile share the input data. The data stored in eDRAM vaults are transferred through routers. After calculation, master tile combines outputs of two tiles.

Hardware Support. As Figure 7(b) shows, we put additional control logic for each RRAM crossbar to support the low power mode. We add a *paring bit* and a *master/salve bit* to indicate the status of the tile. The crossbar array acts as normal when the pairing bit is disabled. When the pairing bit is set, the hot tile is set as master with master/slave bit. The idle

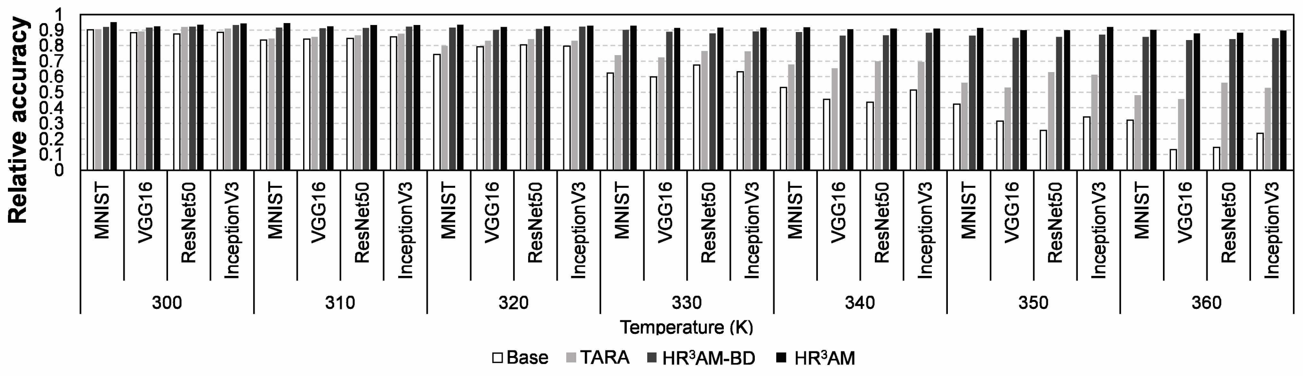

Fig. 8. Inference accuracy of four neural network models.

unit is set as the slave with the same bit. On the master tile, the control logic only enables $2N$ index columns, S&Hs and ADCs are disabled accordingly as well. The slave tile only enables $2N+1$ index columns.

State-of-the-art RRAM architecture [16] does not support dynamic scheduling of tiles. When the pipeline design allocates all the tiles to all the layers, certain tiles are idle during inference but are still bonded to a network layer. Therefore, we reserve several tiles as idle tiles and keep them from being assigned to any network layer. The reserved tiles potentially hurt chip performance due to loss of parallelism. Our performance overhead evaluation in Section IV-D shows throughput decreases only 2.1% on average.

IV. Evaluation

A. Methodology

Our RRAM chip setup is based on a state-of-the-art architecture [16]. Our chip consists of four layers of eDRAM and an RRAM layer. The chip operates at 1.2GHz. We build our own RRAM based neural network accelerator simulator. Our simulator models the RRAM conductance variation between 300K to 400K based on [18]. The simulator updates conductance according to the temperature per 100ms at the crossbar array granularity. Our simulator obtains chip temperatures dynamically from HotSpot [9]. The thermal characteristics of the HMC architecture is obtained from previous work [2]. Our thermal simulation uses a single RRAM cycle (100ns) as the time step as described in Section III-A.

Our simulator coordinates with the Tensorflow framework [1] such that it evaluates any Tensorflow based model on RRAM crossbars. We evaluate our design with a small scale two-layer neural network for MNIST handwritten classification [11] and large scale neural networks (VGG16, ResNet50, and InceptionV3) for ImageNet [6] classification. We use a set of 60000 cases for training and 10000 cases for inference in MNIST. For ImageNet classification networks, we use pretrained weights for convenience. We use 10000 pictures for inference from Large Scale Visual Recognition Challenge [15]. We evaluate the inference accuracy with top-five prediction results. We compare our design with two other baselines:

- **No heat resilience (Base)** implements a plain RRAM chip without any heat resilient design.
- **Temperature-Aware Row Adjustment (TARA)** implements prior work [4] proposed scheme.

Our schemes include:

- **HR^{3}AM with Bitwidth Downgrading (HR^{3}AM-BD) only** implements proposed bitwidth downgrading.
- **HR^{3}AM** implements the both proposed bitwidth downgrading and tile pairing mechanisms.

B. Inference accuracy

Figure 8 shows the inference accuracy of four neural network models under various ambient temperatures. We scale the ambient temperature from room temperature 300K to 360K, where RRAM chip is significantly affected by the poor heat dissipation. We observe several findings from the results. First, our design is resistant to the heat-induced temperature increase. HR^{3}AM-BP and HR^{3}AM manage to respectively obtain at least 83.5% and 87.8% relative accuracy among all the cases. Second, HR^{3}AM has relatively stable accuracy. TARA's accuracy drops by 40.05% on average when the ambient temperature raises 60K. At the same time, the inference accuracy of HR^{3}AM-BP only drops 7.77%, and that of HR^{3}AM only drops 4.82%. Third, networks with larger scale and more layers obtain more benefits from HR^{3}AM. VGG16, ResNet50, and InceptionV3 have a larger improvement from other baselines to our schemes than MNIST. The thermal issue is worse on these networks. These networks have complex structures and process large input images, which lead to intensive data movement between tiles and eDRAM and high power consumption. HR^{3}AM provides further accuracy and thermal optimization for these cases. Overall, we manage to improve inference accuracy by 4.8%–58% over the Base and 4.3%–41.8% over TARA.

C. Thermal status

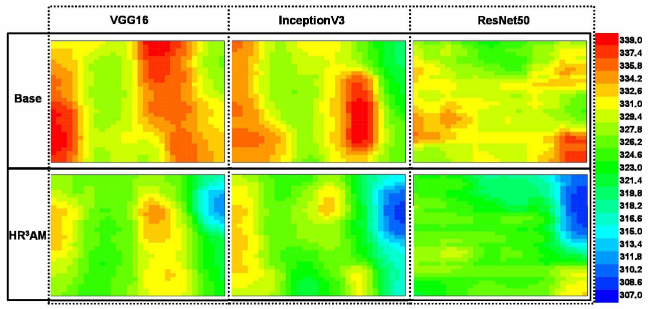

Fig. 9. Comparison of thermal distributions of a RRAM chip when running three networks.

We present the results of thermal status with three network models: VGG16, ResNet50, and InceptionV3. Because TARA [4] does not optimize thermal status, we only compare

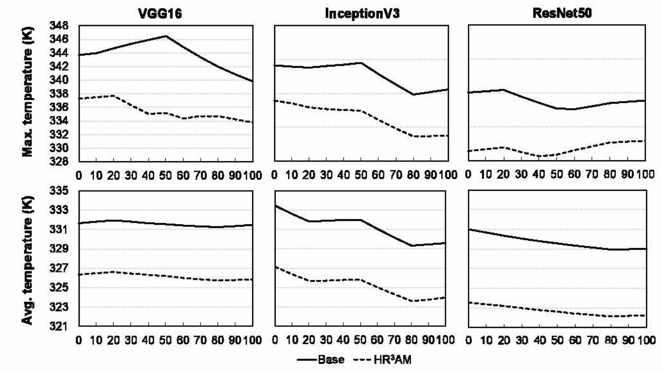

Fig. 10. Comparison of maximum and average temperatures of RRAM chip when running three networks. The x-axis represents the temperature measurement time in milliseconds.

our design with the Base which has no thermal optimization. We reserve 10% tiles as idle tiles for HR^{3}AM.

Figure 9 presents the comparison between HR^{3}AM and the Base on steady-state thermal distribution. We have three observations. First, our design significantly reduces the hot spot region. This indicates that our design manages to identify overheated tiles and mitigate heat for these tiles. Second, idle tiles help mitigate heat and stay at a relatively low temperature. However, these idle tiles form a cold spot. We can further optimize our design by wisely distributing idle tiles across the chip. Lastly, the released heat on hot tiles also benefits the inference accuracy. HR^{3}AM has a smaller percentage of crossbar arrays that triggers bitwidth downgrading than that of the Base, such that it brings a 2% improvement in inference accuracy compared to HR^{3}AM-BP.

To show the effectiveness of HR^{3}AM during chip operation, we measure the maximum and average temperature every 10ms across all the crossbar arrays of the chip. We measure a total of 100ms time during steady operational state. Figure 10 shows the maximum and average temperatures during this time. Our design limits the maximum temperature below 336K and average temperature below 327K. HR^{3}AM manages to contribute a 6.2K decrease on maximum temperature and a 6K decrease on average temperature.

D. Performance and energy

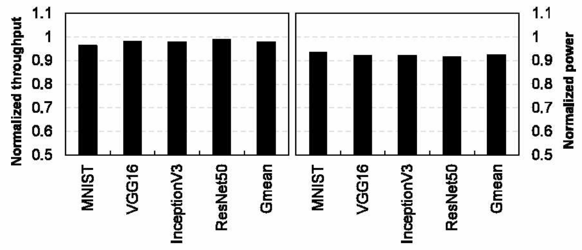

Fig. 11. Normalized (a) throughput and (b) power consumption on HR^{3}AM.

We adopt the performance and power model from [16] to estimate HR^{3}AM. We normalize the results of each network running on HR^{3}AM to the same network running on the Base. Figure 11(a) shows the 2.1% loss on normalized throughput. Both weight encoding in *bitwidth downgrading* and reserving idle tiles in tile pairing cause the performance downgrading. The weight encoding process typically delays the multiplication. However, it only happens when the downgrade bit changes. In tile pairing, HR^{3}AM reserves 10% of the tiles and therefore loses computational parallelism. However, because these tiles work under the low power mode, Figure 11(b) shows a 7.7% average decrease on the normalized power.

V. Conclusion

In this work, we investigate how heat impacts inference accuracy of RRAM based accelerator. Our evaluation shows network inference accuracy drops significantly when the temperature increases. We propose HR^{3}AM, a heat resilient design for RRAM based neural network accelerators. HR^{3}AM consists two mechanisms: *bitwidth downgrading* and *tile pairing*. *Bitwidth downgrading* adjusts weights to the reduced RRAM conductance. Tile pairing matches hot spot RRAM tiles with idle tiles. Compared to state-of-the-art prior work, HR^{3}AM achieves an accuracy improvement by up to 41.8% on four real world applications of CNN models. Our thermal evaluation shows a 6.2K decrease on the maximum temperature and a 6K decrease on average temperature.

VI. Acknowledgment

We thank our shepherd Professor Hai Li for her valuable feedback. This paper is supported in part by NSF grants 1829524, 1829525, 1817077, 1730158, 1527034 and SRC/DARPA Center for Research on Intelligent Storage and Processing-in-memory.

References

[1] M. Abadi et al. TensorFlow: A System for Large-Scale Machine Learning. In *USENIX OSDI*, pages 265–283, Savannah, GA, 2016. USENIX Association.
[2] A. Agrawal et al. Xylem: enhancing vertical thermal conduction in 3D processor-memory stacks. In *MICRO 2017*, pages 546–559. ACM, 2017.
[3] F. Alibart et al. High precision tuning of state for memristive devices by adaptable variation-tolerant algorithm. *Nanotechnology*, 23(7):075201, jan 2012.
[4] M. V. Beigi et al. Thermal-aware Optimizations of reRAM-based Neuromorphic Computing Systems. In *DAC*, pages 39:1–39:6, New York, NY, USA, 2018. ACM.
[5] L. Chen et al. Accelerator-friendly Neural-network Training: Learning Variations and Defects in RRAM Crossbar. In *DATE*, pages 19–24, 3001 Leuven, Belgium, Belgium, 2017. European Design and Automation Association.
[6] J. Deng et al. ImageNet: A large-scale hierarchical image database. In *IEEE CVPR*, volume 00, pages 248–255, June 2018.
[7] K. He et al. Deep residual learning for image recognition. In *IEEE CVPR*, pages 770–778, 2016.
[8] M. Hu et al. Dot-product Engine for Neuromorphic Computing: Programming 1T1M Crossbar to Accelerate Matrix-vector Multiplication. In *DAC*, pages 19:1–19:6, New York, NY, USA, 2016. ACM.
[9] W. Huang et al. HotSpot: A compact thermal modeling methodology for early-stage VLSI design. *IEEE TVLSI*, 14(5):501–513, 2006.
[10] D. Kim et al. Neurocube: A Programmable Digital Neuromorphic Architecture with High-Density 3D Memory. In *ACM/IEEE ISCA*, pages 380–392, June 2016.
[11] Y. LeCun. The MNIST database of handwritten digits. *http://yann.lecun.com/exdb/mnist/*, 1998.
[12] B. Liu et al. Vortex: Variation-aware Training for Memristor X-bar. In *DAC*, pages 15:1–15:6, New York, NY, USA, 2015. ACM.
[13] C. Liu et al. Rescuing memristor-based neuromorphic design with high defects. In *DAC*, pages 1–6, June 2017.
[14] E. Rotem et al. Temperature measurement in the Intel (R) CoreTM Duo Processor. *arXiv preprint arXiv:0709.1861*, 2007.
[15] O. Russakovsky et al. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 115(3):211–252, 2015.
[16] A. Shafiee et al. ISAAC: A Convolutional Neural Network Accelerator with In-situ Analog Arithmetic in Crossbars. In *IEEE/ACM ISCA*, pages 14–26, Piscataway, NJ, USA, 2016. IEEE Press.
[17] K. Simonyan et al. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
[18] C. Walczyk et al. Impact of Temperature on the Resistive Switching Behavior of EmbeddedHfO$_2$-Based RRAM Devices. *IEEE TEC*, 58(9):3124–3131, Sep. 2011.