Learning Assisted Post-Manufacture Testing and Tuning of RRAM-Based DNNs for Yield Recovery

Kwondo Ma, Anurup Saha, Chandramouli Amarnath, Abhijit Chatterjee
School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta GA, USA
kma64@gatech.edu, asaha74@gatech.edu, chandamarnath@gatech.edu, abhijit.chatterjee@ece.gatech.edu

***Abstract*—Variability-induced accuracy degradation of RRAM-based DNNs is of great concern due to their significant potential for use in future energy-efficient machine learning architectures. To address this, we propose a two-step process. First, an enhanced testing procedure is used to predict DNN accuracy from a set of compact test stimuli (images). This test response (signature) is simply the concatenated vectors of output neurons of intermediate and final DNN layers over the compact test images applied. DNNs with a predicted accuracy below a threshold are then tuned based on this signature vector. Using a clustering based approach, the signature is mapped to the optimal tuning parameter values of the DNN (determined using off-line training of the DNN via back-propagation) in a *single step*, eliminating any post-manufacture training of the DNN weights (expensive). The tuning parameters themselves consist of the gains and offsets of the ReLU activation of neurons of the DNN on a per-layer basis and can be tuned digitally. Tuning is achieved in less than a second of tuning time, with yield improvements of over 45% with a modest accuracy reduction of 4% compared to digital DNNs.**

I. Introduction

Among emerging technologies for implementing deep neural networks (DNNs), Resistive Random-Access Memory (RRAM) has gained significant attention as a promising candidate [1] for accelerating DNN computations due to its fast read/write speed, high density and lower power consumption [2]–[5]. However, RRAM-based DNNs are vulnerable to manufacturing process variations which impact their operating reliability and manufacturing yield [6]. Such DNNs may experience significant accuracy degradation, up to 60%, for fully connected and convolutional neural networks, due to RRAM crossbar nonidealities induced by process variations [7]. To characterize these effects, recent studies have proposed circuit-level macro modeling techniques [8] for evaluating memristor-based accelerator performance (classification accuracy). These have been leveraged in prior research to train RRAM based DNNs in ways that factor manufacturing process variations and defects into the training process itself. A variation-aware training (VAT) scheme for addressing hardware limitations and device variations was presented in [9], but is less effective under large process variations. VAT for RRAM crossbar arrays under random variability effects was discussed in [10]. Further, crossbar column rearrangement and weight constrained training was used to resolve crossbar nonidealities in sparse DNNs [11]. However, neither of these methods addresses the effects of *systematic process variations* which have a significant impact on DNN performance [12] (Fig. 1 shows accuracy degradation of MobileNet on CIFAR10 dataset as a function

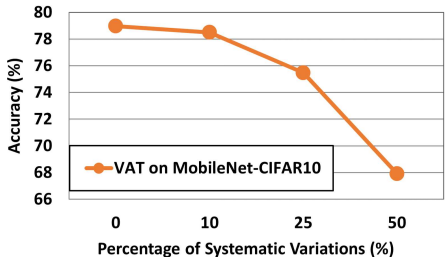


Fig. 1: Accuracy degradation under systematic variations with variability-aware training.

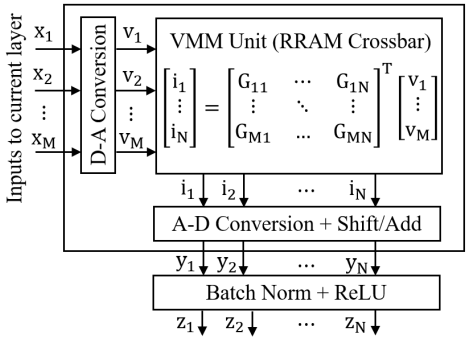
of systematic variations when it is trained in a variability-aware manner for both random and systematic variations). Also, VAT techniques are not effective in enhancing the worst-case performance of DNNs [13]. To address systematic process variability effects, a post-manufacture tuning procedure was proposed in [12] and requires the use of a column of the RRAM crossbar with calibrated weights. Tuning is performed digitally by scaling the input to each neuron by a scaling factor determined by use of the crossbar column above. However, the method is unable to fully recover yield as discussed later in Section V-C. In contrast, the tuning approach developed in this research is performed in the digital domain, offers significantly better yield recovery (see Section V), and can be applied to DNNs with negligible hardware modifications.

The proposed post-manufacture tuning approach consists of two steps: (a) a *testing step* in which a compact subset of test images from the DNN test dataset is applied to the model and the performance of the DNN is predicted from the DNN *test response signature* (consisting of the concatenated vectors of averaged outputs of intermediate layers and outputs of the final dense layer over compact test images applied) and (b) a *tuning step* in which the optimal values of a set of digitally tunable parameters of the DNN are predicted directly (no tuning iterations required) from the same DNN signature. The optimal values above, are determined by a prior off-line back-propagation driven optimization procedure. The key contributions and benefits of the proposed approach are:

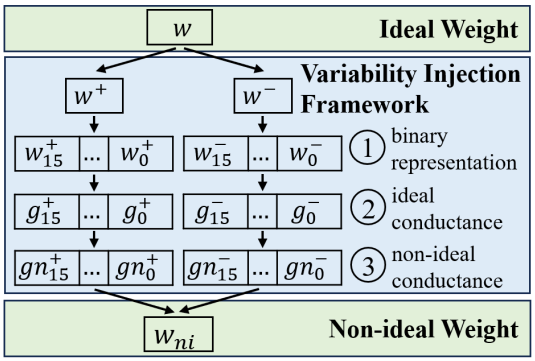
(1) A predictive testing approach is developed which allows determination of the classification accuracy of RRAM-based DNNs that need to be tuned with high precision and low test cost. This minimizes testing and tuning costs in volume manufacturing.

(2) A digital tuning approach for RRAM-based DNNs. Only the ReLU activations of neurons are tuned on a per-layer basis

979-8-3503-4859-0/DATE24/© 2024 EDAA



(a) Computation Flow of Dense Layer



(b) Variability Injection Framework

Fig. 2: Architecture of RRAM-based DNN.

(gain and offset parameters). A DNN with L ReLU layers requires tuning of only $2L$ parameters for yield recovery.
(3) The prediction of the optimal tuning knob values from the DUT response signature is performed by a trained learning algorithm (based on unsupervised response clustering) in a *single* inference pass allowing *testing and tuning for yield recovery to be performed in less than a second* (instead of 10s of seconds or minutes as with other schemes) while achieving accuracy drop within 2% to 4% of a digitally trained DNN.

II. Preliminaries of RRAM

A. Architecture of RRAM-based DNN

In RRAM-based DNN accelerators, vector-matrix multiplication (VMM) is performed using a RRAM crossbar, while pooling, batch normalization and activation operations are carried out using digital computation units. Fig. 2a illustrates the computation flow of a dense layer within the RRAM-based DNN. The real-number inputs to the dense layer are converted to analog voltages using Digital-to-Analog Converters (DACs) and passed on to a VMM unit implemented using RRAM crossbar. The VMM unit performs the multiplication of a stored conductance matrix with the input voltages, resulting in accumulated current outputs. Half of the crossbar columns store positive weights while the remaining columns store negative weights. We use HfO_x based RRAM device [14] with two states: High Resistance State (HRS) representing logic '0' and Low Resistance State (LRS) representing logic '1'. The output currents from the crossbar are converted back to real numbers using Analog-to-Digital Converters (ADCs) and shift/add units. The resulting outputs are then forwarded to the batch normalization and activation functions of the DNN.

B. Variability Modeling Framework

The proposed variability modeling framework quantifies how the weights represented by RRAM devices of a DUT differ from pretrained DNN weights due to process variation. As shown in Fig. 2b, we calculate a non-ideal weight w_{ni} for each ideal weight w of the DNN. Corresponding to each weight w, two weights $w^+, w^- \geq 0$ are calculated as $w^+ = \mathbb{I}[w > 0]|w|$ and $w^- = \mathbb{I}[w < 0]|w|$, which are programmed into the positive and negative sub-arrays of the crossbar respectively ($\mathbb{I}[\cdot]$ denote the Indicator function). Assuming 16-bit fixed point weight representation, the binary representation of w^+ is calculated as $\{w^+_{15}, \cdots, w^+_0\}$ and the j-th bit in the binary representation w^+_j is stored using an RRAM device with conductance g^+_j. We inject variations in the gap dynamics fitting parameter γ^+_j of each RRAM device from its ideal value $\gamma_0 = 16.5$ [14], [15], and calculate γ^+_j as $\gamma^+_j = \gamma_0 + \gamma^{sys} + \gamma^{rand}_j$, where $\gamma^{sys}, \gamma^{rand}_j$ represent the systematic and random components of the variation. Following the variability modeling of [12], the systematic component of variability (γ^{sys}) is sampled once and all RRAM devices in the crossbar share the same γ^{sys}. For all RRAM devices, γ^{rand}_j is sampled independently. The systematic and random components of variation are drawn from two zero mean normal distributions with variance σ^2_{sys} and σ^2_{rand} respectively, and the total variance of the gap dynamics fitting parameter is $\sigma^2_{tot} = \sigma^2_{sys} + \sigma^2_{rand}$. Actual conductance of each RRAM device gn^+_j is calculated as a function of the expected conductance (g^+_j) and the gap dynamics fitting parameter (γ^+_j) using Hspice simulations. Similarly, non-ideal conductances corresponding to all the bits in the binary representation of w^- are calculated as $\{gn^-_{15}, \cdots gn^-_0\}$. We compute an effective conductance $gn_{\text{eff}} = \sum_{j=0}^{15}(gn^+_j - gn^-_j) \times 2^j$ and the effective conductance is mapped back to a non-ideal weight w_{ni}.

III. Predictive Testing of RRAM-based DNNs

The proposed predictive testing technique (also called alternative testing in [15]) is described in Fig. 3. For each of N (statistically significant) RRAM-based DNNs, DNN_i ($1 \leq i \leq N$), selected across diverse process corners, we construct a vector $V_j = [v_{1j}, v_{2j}, ...v_{Nj}]$ corresponding to the j-th image in the testing image dataset, such that $v_{ij} = 1$ if the j-th image is classified correctly by the i-th DNN else $v_{ij} = 0$. The vectors V_j corresponding to all the images in the testing dataset are clustered using agglomerative hierarchical clustering [16] into C clusters and the images corresponding to the medoid of each cluster are selected to be included in the compact test dataset (this is similar to the approach proposed in [15]). The clustering algorithm is used to trade off the number of clusters with cluster size and the euclidean distance between the medoid elements of each cluster. Further, we *mix the down-selected images with randomly filtered images* and repeatedly perform down-selection to generate predictive test image datasets while keeping the size of compact test

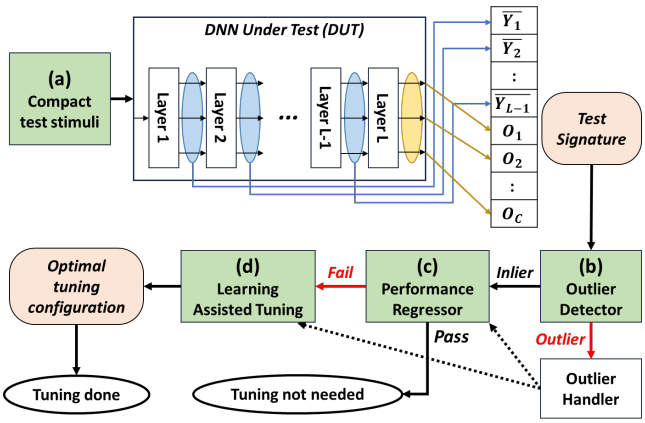


Fig. 3: Post-manufacture testing and tuning framework.

dataset fixed at the prior selected level. Specific filters used include gaussian, median, and box filters along with brightness, contrast, gamma correction and adjustments to hue and color saturation. At each step, the accuracy of the regressor of Fig. 3c is assessed and the compact image dataset which achieves the best inference accuracy prediction is selected for DNN testing. Using experiments on cluster size, we find that only 10 images from the CIFAR-10 dataset of 10,000 testing images (1000× saving) gives adequate test performance of our proposed approach.

Once the compact test dataset is finalized, we observe the averaged outputs of neurons of intermediate layers and raw outputs of the final dense layer of the network under application of the compact test dataset (this is different from the approach of [15]). In Fig. 3, the DNN under test (DUT) has L network layers and C is the number of classes for image classification. The test response from single test image (stimulus) is defined as, $TR = [\overline{Y_1}, \overline{Y_2}, ..., \overline{Y_{L-1}}, O_1, O_2, ..., O_C] \in \mathbb{R}^{L-1+C}$ where $\overline{Y_x}$ represents the averaged output of all the neurons in the x-th intermediate layer. Consequently, when we apply a compact test image dataset of size K, the response signature vector is obtained by concatenating the test response as $\overrightarrow{sig} = [TR_1, TR_2, ..., TR_K] \in \mathbb{R}^{(L-1+C)\times K}$. This signature vector of the DUT is passed to an outlier detector (see Fig. 3b) to assess whether the DNN response signature vector is consistent with the signature vectors of DNNs used to train the regressor of Fig. 3c. In order to identify anomalous DUTs, we employ an Elliptic Envelope (EE) technique [17] to fit the high-dimensional test signature vectors. Outlier devices undergo comprehensive testing using the entire testing dataset within the outlier handler. Additionally, based on the derived inference accuracy, we conduct offline ReLU layer tuning for DUTs that fall below the accuracy threshold. The outcomes of both testing and tuning procedures are subsequently employed to recalibrate the performance regressor and clustering algorithm within the learning-assisted tuning scheme.

The regressor itself is trained to map test signature of DNN under compact image dataset to corresponding DNN classification accuracy values for RRAM-based DNNs, sampled from diverse process corners. The trained multivariate regression spline based regressor (MARS) [18] predicts the classification

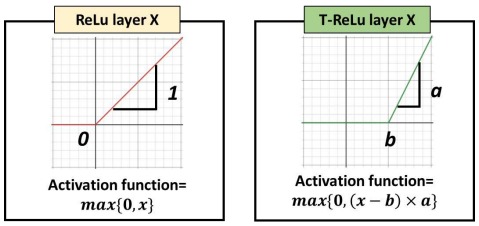


Fig. 4: Trainable-ReLU activation function.

accuracy of a tested DNN as $\widehat{a}$, using the derived signature $\overrightarrow{sig}$ above. We consider the accuracy prediction's confidence level, taking into account the inherent prediction error ($err = |\widehat{a} - a|$), where a denotes the actual DNN accuracy. The error in accuracy prediction is assumed to have a gaussian distribution such that if the predicted accuracy of a specific DNN exceeds the desired accuracy threshold by an acceptable confidence margin, $\widehat{a} \geq a_{th} + k\sigma$, it is categorized as a good DNN. All other DNNs are subjected to post-manufacture tuning. The proposed predictive testing reduces the standard deviation of prediction error of the RRAM-based DNN accuracy from 1.37 for the scheme of [15] to 0.92 for MobileNet on the CIFAR10 dataset. This reduction of prediction error results in lower tuning costs during manufacturing.

IV. Post-Manufacture Tuning

We introduce a novel post-manufacture tuning method for RRAM-based DNNs, digitally adjusting the gain and offset parameters of ReLU activation functions of the DNN to restore accuracy. This is augmented by a learning assisted tuning scheme using test signature clustering to discover optimal tuning configurations for DUTs in a single step.

A. Digital ReLU Layer Tuning

We propose a digital ReLU layer tuning technique to address process variability effects in RRAM-based DNNs. This method involves transforming standard ReLU activation functions within the DNN model into proposed trainable ReLU functions with two global tuning parameters per layer. Fig. 4 illustrates the standard ReLU function, where the x-intercept is 0 and the slope is 1. Our tuning scheme introduces the 'T-ReLU' function, which incorporates a trainable offset (b) and gain (a) as layer tuning parameters. Thus, the T-ReLU activation function is defined as, T-ReLU$(x) = \max(0, (x - b) \times a)$.

Algorithm 1 performs optimization of the cross-entropy loss function using the Adam optimizer [19] for an assigned learning rate (line 1). For each DUT that requires tuning, we transform all L ReLU activation layers in the network into T-ReLUs and freeze all weights in the DUT except for the tuning parameters associated with these T-ReLUs (line 2-5). Here, p represents the set of tuning parameters of our optimization, containing the vectors of gain and offset for all T-ReLU layers. For each training epoch, inference is performed using the training dataset applied to the DUT, making predictions based on the current T-ReLU configuration (line 7). The loss is then calculated by comparing these predictions with the training labels (line 8). For each layer in reverse order, we calculate the loss gradient with respect to the layer's T-ReLU

Algorithm 1 Optimization of T-ReLU tuning parameters

1: Assign learning rate (η), Loss function ($\mathcal{L}$), Optimizer ($\mathcal{F}$)
2: **for** DUT that requires a tuning **do**
3: Transform L number of ReLU activation functions to T-ReLU
4: Freeze all the weights except the set of tuning parameters ($p = \{\overrightarrow{a}, \overrightarrow{b}\}$) of T-ReLU activation functions
5: **end for**
6: **for** epoch = 1, 2, ... **do**
7: Perform inference under the stimuli from training dataset
8: Calculate loss $\mathcal{L}$ between DUT outcomes and true labels
9: **for** l = L, $L-1$, ... , 1 **do**
10: Calculate the gradient with respect to p: $\frac{\partial \mathcal{L}}{\partial p^{(l)}}$
11: Update $p^{(l)}$ using Optimizer: $p^{(l)} = \mathcal{F}(p^{(l)}, \frac{\partial \mathcal{L}}{\partial p^{(l)}}, \eta)$
12: **end for**
13: **end for**
14: **return** Optimized tuning parameters p^*

tuning parameters and update them using the optimizer (line 9-12). After training, the algorithm returns the optimized tuning parameters of the T-ReLU functions for the DUT (line 14).

Since the computation of activation functions in RRAM-based DNNs occurs in the digital domain, we can seamlessly integrate the tuning parameters into the network with almost zero hardware modification. Moreover, our method only requires *a pair of tuning parameters for each ReLU layer*, effectively capturing the perturbations caused by systematic variability effects. Our proposed digital ReLU layer tuning thus significantly reduces the tuning cost, time, and parameter complexity compared to traditional weight retraining.

Tuning on smaller training subset: The efficiency of the proposed digital ReLU layer tuning approach can be enhanced by utilizing a smaller training subset instead of the entire training dataset. This approach involves *a tradeoff between the training cost and the model performance.* We refer to this method as 'subset tuning', where the size of the training data subset used is represented as a proportion of the complete dataset. For the CIFAR-10 dataset, using 5,000 train images for subset tuning would be referred to as '10% subset tuning' (considering the entire set of 50,000 training images).

B. Learning Assisted Tuning

The learning assisted tuning technique is based on the observation that the signatures obtained from predictive testing contain enough information about the statistics of RRAM-based DNNs for successful prediction of model accuracy. Algorithm 2 outlines a learning assisted approach (as shown in Fig. 3d) to tuning new DUTs without offline back-propagation. This approach significantly reduces the computational complexity and effort required for tuning new DUTs while leveraging the knowledge gained from previous tuning experiments.

Algorithm 2 begins by collecting the optimized tuning parameters from previously post-tuned sample DUTs that have undergone our proposed ReLU tuning process (line 1). Then, we apply agglomerative hierarchical clustering [16] to cluster the signature vectors of these post-tuned DUTs. The signature vectors capture essential characteristics of each DUT's tuning configuration. For each new DUT that requires online tuning, we iterate through every cluster in the signature vector

Algorithm 2 Learning Assisted Tuning

1: Collect optimized tuning parameters of post-tuned sample DUTs
2: Cluster the signature vectors of these DUTs
3: **for** New DUT that requires tuning **do**
4: **for** Every cluster in signature vector space **do**
5: Calculate the distance between its medoid and $\overrightarrow{sig}$ of DUT
6: **end for**
7: Select the cluster which has the minimum distance
8: Find the nearest DUT* within this cluster with new DUT
9: **end for**
10: **return** Tuning parameters of DUT*

space and calculate the distance between the medoid of the cluster and the signature vector of the new DUT (lines 4-6). This distance represents how closely the new DUT's tuning requirements match those of the clusters. The cluster with the minimum distance is thus considered as the best match for the tuning needs of the new DUT. Within this selected cluster, we find the nearest tuned DUT to the new DUT based on their signature vector distances. The algorithm then returns the tuning parameters of this nearest tuned DUT. These tuning parameters are then applied to the new DUT without the need for expensive back-propagation.

V. Results

This section presents experimental validation of our post-manufacture testing and tuning approach on various RRAM-based DNNs and compares it to the state of the art.

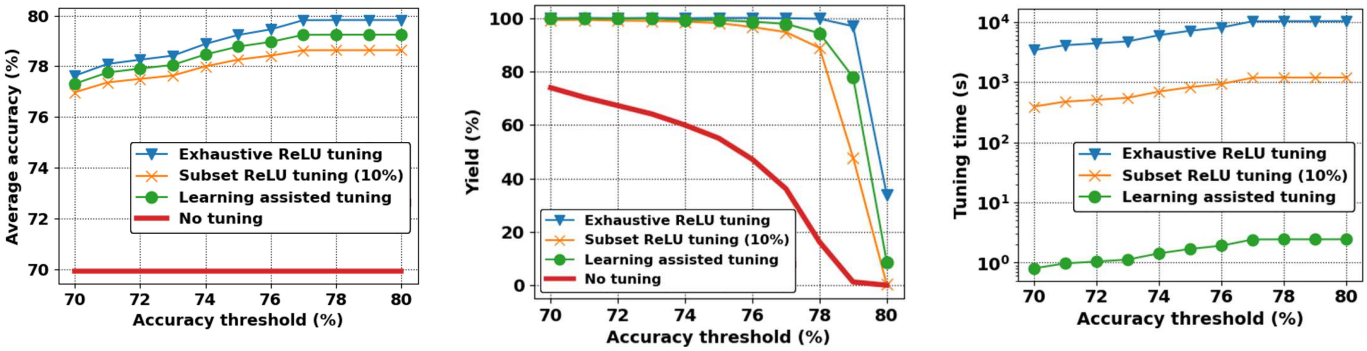
A. Experimental Methodology

Table I provides the datasets and architectural details (number of convolutional layers (#Conv), fully connected layers (#FC), ReLU activation layers (#ReLU), and the ideal accuracy (Ideal Acc)) for the DNNs tested. The ideal (reference) accuracy represents the classification accuracy of the models with ideal weights in the digital domain. The variability modeling framework is derived from HSPICE using the PTM model [23] at 65 nm CMOS technology. We assume 50% contribution from systematic and 50% contribution from random variability components [12], [15], i.e., $\sigma_{sys} = \sigma_{rand} = \sigma_{tot}/\sqrt{2}$. We set σ_{tot} between 3.5% to 4.8% of γ_0 to generate perturbed RRAM-based DNNs. For all benchmark DNN applications, we use PyTorch for simulation using 16-bit fixed-point precision.

The variability modeling framework was used to generate 2000 DNNs across diverse process corners via statistical sampling of the process space. 1000 of these devices were used for test stimulus generation, training the outlier detector, training the regressor for predicting DNN accuracy, and storing the post-tuning data of the sampled DNNs for learning assisted tuning. The other 1000 DNNs were utilized as the baseline for evaluating yield loss due to process variations. We subjected these DUTs to the predictive testing methodology while

TABLE I: Benchmark DNN applications

Dataset	Network	#Conv	#FC	#ReLU	Ideal Acc
CIFAR-10	MobileNet [20]	27	1	27	83.53 %
	VGG16 [21]	13	1	13	93.24 %
CIFAR-100	ResNet18 [22]	17	1	17	75.95 %
	VGG16 [21]	13	3	15	72.36 %



(a) Average accuracy (b) Yield (c) Tuning time

Fig. 5: Evaluation results of MobileNet on CIFAR-10 dataset.

employing three different tuning techniques for comparison: (a) Digital ReLU layer tuning utilizing the complete training dataset, referred to as 'exhaustive ReLU tuning,' (b) Digital ReLU layer tuning using only a randomly selected 10% subset of the training dataset, and (c) Learning assisted (LA) tuning leveraging the optimization results gained from tuning on sampled DUTs.

B. Evaluation of Post-Manufacture Testing and Tuning

Fig. 5 shows the average accuracy of DNN devices (RRAM-based MobileNet on CIFAR-10) after post-manufacture tuning (all devices after the entire testing-tuning process of Fig. 3) with different acceptance accuracy thresholds for tuning. The average accuracy of the perturbed DNNs on the RRAM crossbar without tuning stands at 70%, resulting in an accuracy loss of 13.5% (indicated by the solid line in Fig. 5a). Since no tuning is performed, this is flat across all accuracy thresholds. Three other plots for average accuracy vs. acceptance accuracy threshold are shown: (a) exhaustive ReLU tuning (triangle markers) in which all 50,000 training images are used to train only the ReLU parameters of the DNN, (b) learning assisted tuning (circle markers) and (c) tuning with only a randomly selected 10% of the CIFAR-10 training dataset consisting of 5000 images used to train only the ReLU parameters of the DNN. As the acceptance accuracy threshold increases, the average accuracy also rises because devices below the threshold are tuned. The results of 10% subset-tuning showcase the feasibility of our proposed ReLU tuning approach, even with a smaller training dataset.

The yield of a DNN manufacturing process is the (number of DNNs above a specified *threshold of classification accuracy*)/(total number of DNNs manufactured). The yield of DNNs without any tuning, rapidly decreases with increasing accuracy thresholds, as shown by the solid line of Fig. 5b . We see that more than 70% of RRAM-based DNNs are deemed good with up to 14% drop in accuracy allowed, but stricter testing criteria (up to 4% accuracy drop) limits the percentage of good DUTs to just 2%. All three tuning schemes demonstrate nearly perfect yield improvement for acceptance accuracy threshold $< 72\%$). For comparison, in the high accuracy scenario with a 78.5% threshold, exhaustive ReLU tuning (triangle markers) is the most effective, reaching 99.1% yield. It is followed by learning assisted tuning (circle markers) at 89.4% yield. 10% subset-tuning (X markers) achieves 77.3% yield.

Fig. 5c, shows the tuning time (measured as the number of GPU clock cycles required for optimizing T-ReLU tuning parameters through post-manufacture back-propagation) vs. acceptance accuracy threshold for the three tuning schemes discussed above. For learning assisted tuning, the CPU computation time for clustering based inference was assessed. As the accuracy threshold increases, the DNNs require increased tuning time. The 10% subset ReLU tuning (X markers) exhibits an 8.7× speedup in comparison to exhaustive ReLU tuning (triangle markers). Furthermore, our proposed learning assisted tuning (circle markers) achieves an exceptional 4245× speedup compared to exhaustive ReLU tuning, while incurring a minimal compromise of less than 4% in DNN yield. Note that the evaluation of testing and tuning on VGG16 model shows comparable results; thus we omit them for brevity.

C. Comparison with the State of the Art

Fig. 6 illustrates the yield of ResNet18 on CIFAR-100 with respect to the acceptance accuracy threshold for exhaustive ReLU tuning (triangular markers), learning assisted tuning (circle markers), the broad training and tuning approach of [12] (cited as VAT + ST in Fig. 6 with X markers) and no tuning. VAT incorporates both random and systematic variabil-

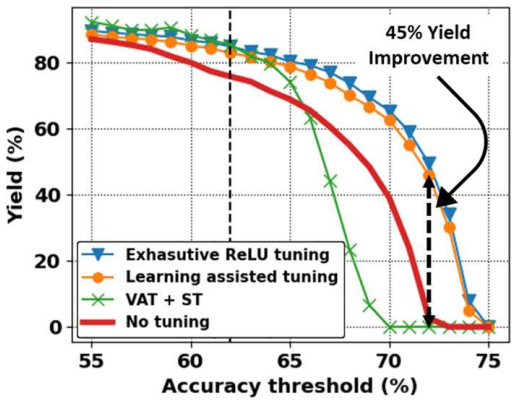


Fig. 6: Yield comparison of ResNet18 on CIFAR-100.

TABLE II: Comparison with the state of the art

	Post-manufacture retraining	Variability model[a]	Tuning time	Yield (8% Acc drop)	Test
ERT[b]	2 params per layer	Ran+Sys	$< 20s$	73.6%	Yes
LAT[b]	None	Ran+Sys	$< 1s$	70.0%	Yes
[12]	None	Ran+Sys	$< 1s$	23.4%	No
[24]	0.5−5% of weights	Ran	Large	Not reported	No

a. Ran and Sys represent random and systematic process variations, respectively.
b. LAT and ERT stand for learning assisted and exhaustive ReLU tuning, respectively.

ity during DNN training. VAT + ST refers to application of the self-tuning (ST) technique described in [12] to the DNN obtained after variability aware tuning. VAT + ST can partially recover the yield loss of DNNs for accuracy threshold $< 62\%$. However, the yield of VAT + ST becomes worse than RRAM-based DNNs without any tuning for accuracy threshold $> 66\%$. In contrast, our proposed learning assisted tuning scheme achieves yield of more than 60% for accuracy threshold of 70%, where the VAT + ST technique becomes completely ineffective. Furthermore, our tuning scheme achieves 45% yield improvement when the accuracy threshold is 72% (allowable accuracy drop from ideal model = 4%) which is important for safety-critical applications.

Table II provides a qualitative comparison of our proposed tuning methods, referred to as Exhaustive ReLU Tuning (ERT), and Learning Assisted Tuning (LAT), with two state-of-the-art techniques for addressing the challenges posed by manufacturing process variations in RRAM-based DNNs. In CorrectNet [24], the effects of random variations are studied, and post-manufacture re-training of $0.5 - 5\%$ of the network weights is performed. As per our own experiments, the tuning time of this approach (the time taken to retrain the specific weights using back-propagation) is relatively large when compared to the proposed exhaustive ReLU tuning, which takes less than 20 seconds of tuning time and focuses on retraining only two ReLU parameters per layer of the network. Both [12] and our proposed learning assisted tuning do not require a retraining process to recover model performance, and their online tuning time is less than a second. However, when evaluating manufacturing yield with an 8% allowable accuracy drop, [12] achieves a modest yield of 23.4%. In contrast, the proposed Exhaustive ReLU Tuning method (ERT) achieves yield of 73.6%, while Learning Assisted Tuning (LAT) achieves 70.0% yield. Furthermore, our proposed tuning schemes incorporate a compact testing procedure prior to post-manufacture tuning, which is necessary for reducing testing and tuning costs, and is not considered in prior research.

VI. Conclusion

This paper presented and validated a two-step approach to address the impact of process variability in RRAM-based DNNs. It consists of regressor-based testing to predict DNN accuracy from image subsets and post-manufacture tuning using layer-by-layer adjustment of ReLU activation parameters. An energy-efficient learning-assisted tuning scheme is also presented and validated as a substitute for post-manufacture tuning. Our approach achieves more than 45% yield improvement with an accuracy drop of 4% compared to ideal digital DNNs, outperforming the state-of-the-art.

Acknowledgment

This research was supported by the U.S. National Science Foundation under Grant: 2128149.

References

[1] M. Prezioso *et al.*, "Training and operation of an integrated neuromorphic network based on metal-oxide memristors," *Nature*, vol. 521, no. 7550, pp. 61–64, 2015.

[2] S. Yu *et al.*, "Emerging memory technologies: Recent trends and prospects," *IEEE Solid-State Circuits Magazine*, vol. 8, no. 2, pp. 43–56, 2016.

[3] M. Hu *et al.*, "Memristor-based analog computation and neural network classification with a dot product engine," *Advanced Materials*, vol. 30, no. 9, p. 1705914, 2018.

[4] A. Shafiee *et al.*, "Isaac: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars," in *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, 2016, pp. 14–26.

[5] G. W. Burr *et al.*, "Experimental demonstration and tolerancing of a large-scale neural network (165 000 synapses) using phase-change memory as the synaptic weight element," *IEEE Transactions on Electron Devices*, vol. 62, no. 11, pp. 3498–3507, 2015.

[6] S. Jain *et al.*, "Rxnn: A framework for evaluating deep neural networks on resistive crossbars," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 40, no. 2, pp. 326–338, 2020.

[7] I. Chakraborty *et al.*, "Technology aware training in memristive neuromorphic systems for nonideal synaptic crossbars," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 5, pp. 335–344, 2018.

[8] S. Roy *et al.*, "Txsim: Modeling training of deep neural networks on resistive crossbar systems," *IEEE Transactions on Very Large Scale Integration(VLSI) Systems*, vol. 29, no. 4, pp. 730–738, 2021.

[9] B. Liu *et al.*, "Vortex: Variation-aware training for memristor x-bar," in *Proceedings of the 52nd Annual Design Automation Conference*, 2015, pp. 1–6.

[10] Y. Long *et al.*, "Design of reliable dnn accelerator with un-reliable reram," in *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2019, pp. 1769–1774.

[11] A. Bhattacharjee *et al.*, "Examining and mitigating the impact of crossbar non-idealities for accurate implementation of sparse deep neural networks," in *2022 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2022, pp. 1119–1122.

[12] Z. Deng *et al.*, "Variability-aware training and self-tuning of highly quantized dnns for analog pim," in *2022 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2022, pp. 712–717.

[13] Z. Yan *et al.*, "Computing-in-memory neural network accelerators for safety-critical systems: Can small device variations be disastrous?" in *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design*. IEEE/ACM, 2022, pp. 1–9.

[14] P.-Y. Chen *et al.*, "Compact modeling of rram devices and its applications in 1t1r and 1s1r array design," *IEEE Transactions on Electron Devices*, vol. 62, no. 12, pp. 4022–4028, 2015.

[15] K. Ma *et al.*, "Efficient low cost alternative testing of analog crossbar arrays for deep neural networks," in *2022 IEEE International Test Conference (ITC)*. IEEE, 2022, pp. 499–503.

[16] W. H. Day *et al.*, "Efficient algorithms for agglomerative hierarchical clustering methods," *Journal of classification*, vol. 1, no. 1, pp. 7–24, 1984.

[17] P. J. Rousseeuw, "Least median of squares regression," *Journal of the American statistical association*, vol. 79, no. 388, pp. 871–880, 1984.

[18] J. H. Friedman, "Multivariate adaptive regression splines," *The annals of statistics*, vol. 19, no. 1, pp. 1–67, 1991.

[19] D. P. Kingma *et al.*, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[20] A. G. Howard *et al.*, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

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

[22] K. He *et al.*, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[23] Nanoscale Integration and Modeling (NIMO) Group, ASU, "Predictive Technology Model," http://ptm.asu.edu, 2011, online; accessed 8 April 2022.

[24] A. Eldebiky *et al.*, "Correctnet: Robustness enhancement of analog in-memory computing for neural networks by error suppression and compensation," in *2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2023, pp. 1–6.