Offline Training-Based Mitigation of IR Drop for ReRAM-Based Deep Neural Network Accelerators

Sugil Lee, Mohammed E. Fouda, Jongeun Lee, *Member, IEEE*, Ahmed M. Eltawil, *Senior Member, IEEE*, and Fadi Kurdahi, *Fellow, IEEE*

***Abstract*—Recently, resistive RAM (ReRAM)-based hardware accelerators showed unprecedented performance compared the digital accelerators. Technology scaling causes an inevitable increase in interconnect wire resistance, which leads to IR drops that could limit the performance of ReRAM-based accelerators. These IR drops deteriorate the signal integrity and quality, especially in the crossbar structures which are used to build high-density ReRAMs. Hence, finding a software solution, which can predict the effect of IR drop without involving expensive hardware or SPICE simulations, is very desirable. In this article, we propose two neural networks models to predict the impact of the IR drop problem. These models are used to evaluate the performance of the different deep neural network (DNN) models including binary and quantized neural networks showing similar performance (i.e., recognition accuracy) to the golden validation (i.e., SPICE-based DNN validation). In addition, these predication models are incorporated into the DNN training framework to efficiently retrain the DNN models and bridge the accuracy drop. To further enhance the validation accuracy, we propose incremental training methods. The DNN validation results, done through SPICE simulations, show very high improvement in performance close to the baseline performance, which demonstrates the efficacy of the proposed method even with challenging datasets, such as CIFAR10 and SVHN.**

***Index Terms*—Binary neural network, deep neural network (DNN), IR drop, quantized neural network, resistive RAM (ReRAM) crossbar array (RCA), variability.**

I. Introduction

WHILE Resistive RAM (ReRAM) crossbar arrays (RCAs) are considered as one of the most promising technologies for highly efficient neural network hardware, much of its promise critically depends on the assumption that RCAs can function as a computing unit as well as a storage unit [2]. In an ideal condition, an RCA can do parallel matrix–vector multiplication (MVM) between a *weight* conductance matrix and an *input* voltage vector, with the time complexity of $O(1)$ instead of $O(n^2)$. RCAs can not only provide such an extremely parallel matrix operation but also eliminate the von Neumann bottleneck since the computation is done right where the weight matrix is stored. For neural network hardware that spends most of its energy doing matrix multiplications, RCA could be an ideal technology, solving both computation and communication problems simultaneously with radically improved efficiency. Furthermore, a passive RCA, one that consists of memristors only without active transistors (i.e., 0T1R), has superior area advantage.

However, if an RCA cannot function as a computing unit, for instance, due to its MVM result being unrecoverably distorted, then an RCA would degenerate into a memory, annulling the great promises mentioned above. Such is the threat that can be caused by IR drop when the array size is large or wire resistance increases relative to low-resistance state (LRS) resistance. Since the size of RCA arrays is expected to grow, and device-to-wire resistance ratio depends on many factors, including material choice, it is important to address the IR drop problem at the system level as much as possible.

Recently, using a comprehensive crossbar-based simulation framework, He *et al.* [3] showed that IR drop can cause as much as 66 percentage points (pps) drop in classification accuracy for LeNet-5 on MNIST when using small RCAs (64×64), which will worsen with larger RCAs or more challenging datasets. They also proposed a method, NIA, based on random noise for training networks without expensive IR drop simulation. Another method, called the *mask* method [4], was shown to recover near-baseline classification accuracy for a binarized neural network (BNN) on MNIST when using larger RCAs (128×128). However, for more challenging datasets such as CIFAR10, it turns out that the IR drop problem is very difficult to mitigate even with BNNs, not to mention the challenges of larger arrays and higher wire resistance, casting doubt on the feasibility and scalability of the RCA-based neural network hardware approach.

In this article, we propose a novel method for passive RCAs, which can learn as well as recall the distortion pattern of MVM computation for any given input based on statistical machine learning, in a fast and reliable manner.

Manuscript received 25 August 2021; revised 31 January 2022 and 12 April 2022; accepted 10 May 2022. Date of publication 23 May 2022; date of current version 20 January 2023. This work was supported in part by IITP through Artificial Intelligence Graduate School Program (UNIST) under Grant 2020-0-01336, through Neuromorphic Computing Software Platform for Artificial Intelligence Systems under Grant 1711080972, and through ITRC Support Program under Grant IITP-2021-0-02052; in part by NRF Grant funded by the MSIT of Korea under Grant 2020R1A2C2015066; and in part by the Free Innovative Research Fund of UNIST under Grant 1.170067.01. The EDA tool was supported by the IC Design Education Center (IDEC), South Korea. This paper extends an earlier version presented at DAC '20 [1], with support for quantized neural networks and more experimental results and analyses [DOI: 10.1109/dac18072.2020.9218735]. This article was recommended by Associate Editor L.-C. Wang. *(Corresponding author: Jongeun Lee.)*

Sugil Lee and Jongeun Lee are with the Department of EE, Ulsan National Institute of Science and Technology, Ulsan 44919, South Korea (e-mail: jlee@unist.ac.kr).

Mohammed E. Fouda and Fadi Kurdahi are with the Department of EECS, University of California at Irvine, Irvine, CA 92697 USA.

Ahmed M. Eltawil is with the Division of CEMSE, King Abdullah University of Science and Technology, Thuwal 23955, Saudi Arabia.

Digital Object Identifier 10.1109/TCAD.2022.3177002

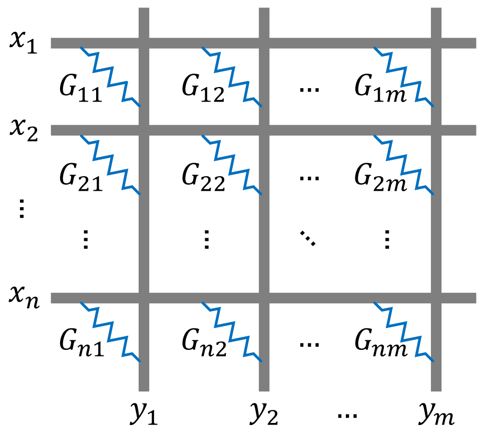


Fig. 1. Performing MVM computation on a passive RCA.

Our method enables greatly enhanced training of RCA-based neural networks, significantly increasing the range of usable device-to-wire resistance ratios and RCA sizes.

In this article, we make the following contributions.

1) We interpret the IR drop problem in passive RCA-based MVM computation as a function identification problem, making it accessible to a number of known techniques, such as regression and machine learning.
2) We propose two prediction models including a convolutional neural network (CNN) to accurately learn and predict the distortion pattern caused by IR drop.
3) We propose a novel *incremental training* method that can improve the training performance of deep neural networks (DNNs) under large wire resistance values.
4) We show empirically that our training method is effective in mitigating the effect of IR drop in passive RCA-based neural network hardware for various RCA sizes and wire resistance values.

This article is organized as follows: Section II explains the IR drop problem and the previous works to mitigate the impact of the problem on the DNNs performance. Section III-A discusses the proposed methodology on training DNNs. The proposed IR drop predication methods are then discussed in Section IV. Finally, Section VI discusses our experimental setup and validation framework and results.

II. Background and Previous Work

A. IR Drop in ReRAM Crossbar Array

Fig. 1 illustrates a passive RCA of size $n \times m$ performing an MVM operation between a weight matrix W of size $m \times n$ and an n-dimensional input vector x, generating an m-dimensional output $y = Wx$, where x is given as voltage, y in current, and W is programmed as conductance after transpose ($G = W^T$).

This works only if the voltage across the two terminals of a ReRAM cell labeled with G_{ij} is exactly x_i. In a crossbar structure, it is usually not the case. For instance, if the wire has resistance, the input voltage x_i is divided between the wire and ReRAM cells, generating less current than $G_{ij}x_i$. Naturally, the farther the ReRAM cell is from input/output ports, the higher the IR drop.

The amount of IR drop depends on ReRAM cells' resistance, which is programmable, as well as wire resistance. Thus, the single most important architecture parameter determining the overall severity of IR drop is the LRS-to-wire resistance ratio. But to find out the exact amount of output current, one needs SPICE simulation.

In addition to device faults and variation, ReRAM resistance may drift due to stochastic noise, which, in turn, affects the amount of IR drop. Therefore, all these effects must be considered together, which requires a huge amount of computing resources (on the order of days with a capable workstation), even to evaluate the *inference* accuracy of an MNIST-level neural network.

B. Target Application

Our target application is RCA-based neural network hardware, that is, a hardware accelerator for neural networks where matrix multiplication is performed on RCAs [5]. A large matrix multiplication is divided into smaller ones (via loop tiling), each of which is mapped to an RCA, with necessary input/output routing circuitry added using CMOS technology. All RCAs are dedicated, i.e., completely parallel, since programming ReRAM is costly. Convolution operation can be implemented using matrix multiplication [6].

Quantized neural networks (QNNs) often have batch normalization (BN) layers [7]. BN consists of shift and scaling operations, and help training converge faster. BN is crucial, not just for BNNs but for all low-precision networks. The problem with BN is that the computation is a bit complicated, including a division operation. However, for QNNs with binary activation, we can actually eliminate BN layers by modifying the bias values of the preceding layer [8]. Note that the bias values in BNNs, whether before or after BN elimination, does not require very high precision, and much less than in multibit networks.[1]

C. Related Work

Several ReRAM-based neural network architectures have been proposed [5], [9], [10], which are all based on MVM operation on RCA. ISAAC [5], for instance, carefully optimizes the system throughput and cost while supporting multibit weight and activation. However, the RCA output, which is analogue, needs to be converted to digital through costly ADC, which is shared and becomes a performance bottleneck. Some ReRAM-based neural network architectures [11], [12] support training as well as inference using ReRAM arrays. However all these architectures assume perfect MVM operation, and do not address the training problem in the presence of distorted MVM operation such as when using passive RCAs for superior area advantage.

The consideration of IR drop needs to be done in both inference and training of passive RCA-based neural networks. He *et al.* [3] presented one such framework, with two findings. First, the accuracy drop can be severe: 99% classification accuracy when IR drop is ignored, plummets to mere 32% when IR drop is considered, for the LeNet-5 network with

[1]Since only the sign of the output activation needs to be determined, the bias only needs to be $\lceil \log_2 n \rceil$-bit, where n is the number of terms added in the computation of output activation.

the MNIST dataset, using 64 × 64 RCAs. Second, the RCA size is a critical parameter; in the same setting, when the RCA size is 32 × 32, the accuracy drop is only about 3%.

They also propose NIA, which is to model, as Gaussian noise, RCA *output current's shift* via IR drop-enabled inference simulation, and then generate the same Gaussian noise to be added to RCA outputs during training. This way, NIA can avoid costly IR drop simulation (during training) while training a network with IR drop effect taken into account.

There are two problems with this method. First, using an *additive* term to model the effect of IR drop may not be right. Second, the IR drop pattern is inherently 2-D, varying across rows and columns of an RCA, which is ignored by focusing on the output of RCA, which is 1D.

The mask method [4] captures the IR drop pattern as a *2-D matrix* that has the same size as the RCA. The mask is *multiplied* to a weight matrix in an elementwise manner before MVM computation. The mask method works because the IR drop pattern has a high spatial correlation, which is captured by mask.

Machine learning-based methods [13]–[15] to predict the effect of IR drop is mentioned by a survey paper [16]. Ho and Kahng [13] proposed an incremental method to predict and fix violations called IncPRID, consisting of feature extraction, IR drop prediction, and incremental design modification steps. XGBIR [14] uses an ensemble tree and features extracted from a power grid to estimate the region that suffers from IR drop. PowerNet [15] is a CNN-based method with some pre- and postprocessing. However, all these methods target power delivery network (PDN) applications and, as such, their objective is to prevent IR drop by manually modifying PDN design, whereas we do not modify RCA hardware but try to reclaim RCA hardware despite IR drop by simply adjusting weight parameters.

Huang *et al.* [17] proposed to add a tunable RRAM row to an RCA to compensate for output current deviation. However, for binary RCA adding one row is equivalent to adding either 0 or 1, the effectiveness of which is very limited. Zhu *et al.* [18] proposed hardware and software methods based on the empirical observation of variation patterns due to IR drop, demonstrating the effectiveness for the MNIST classification task. Roy *et al.* [19] presented a nonideality modeling framework for the purpose of on-chip training on RCA hardware.

Apart from output error mitigation, Chakraborty *et al.* [20] proposed a method (GENIEx) to predict the output distortion in the presence of both IR drop and I-V nonlinearity. However, it does not consider nonideality-aware training of target DNNs.

III. Training Under Distorted MVM

The problem we deal with in this article is to find the best weight of a DNN in the presence of MVM distortion such as IR drop.

A. Methodology

Fig. 2 illustrates our experimental methodology. The first step is *baseline training*. The weight obtained from baseline

Step 1: $W_1 \leftarrow$ Baseline training
Step 2: Validate(W_1)
Step 3: $W_2 \leftarrow$ Train under distorted MVM
Step 4: Validate(W_2)

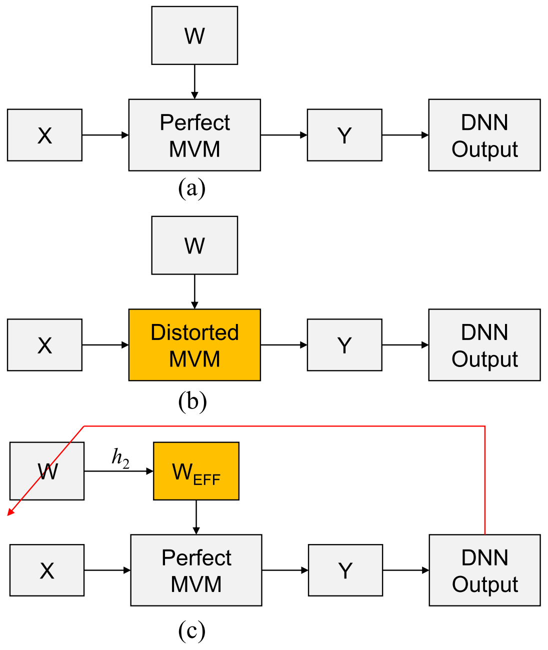


Fig. 2. Our experimental methodology. (a) Baseline training: weight optimized for perfect MVM. (b) Validation: inference using distorted MVM (e.g., SPICE). (c) Example of training under distorted MVM.

training, if programmed to an accelerator that has perfect MVM computation (e.g., digital CMOS), should give the identical accuracy as in software inference. In this case, $Y = WX$, where Y is the MVM output, W is the weight matrix, and X is the input activation.

Second, if we apply the weight from the baseline training to an RCA-based accelerator, the distorted MVM will cause poor accuracy. In this case, Y is still a function of W and X but not WX; i.e., $Y = g(W, X) \neq WX$. To find g out, we need simulation such as SPICE, which takes long but is feasible if we do inference only. This step is referred to as *validation*, since if the accuracy in this setup is shown to be high enough, it confirms that the weight has been correctly optimized under MVM distortion.

Third, the validation setup is not useful for training due to two reasons: 1) running IR drop simulation inside a training loop is extremely time consuming and 2) such simulation is not differentiable; thus, no guarantee on convergence. Instead, our novel approach is to use a *surrogate model* that can predict the outcome of the distorted MVM in a fast and differentiable way, so that we can use the model for training. Once a suitable model for g is found, it is substituted for distorted MVM computation (i.e., SPICE simulation), and we run training again to find a new W under MVM distortion.

Finally, we run validation again to see if the weight from step 3 gives good accuracy when using the real distorted MVM such as SPICE simulation. Note that there can be quite a difference between the (test) accuracy evaluated with the surrogate model (i.e., accuracy from step 3), and the validation accuracy obtained in step 4. This is because the surrogate model may not agree with SPICE simulation for every input combination. The validation accuracy from step 4 is the main evaluation metric we use in this article.

B. Linearity Principle

While the number of ways to model distorted MVM computation is unbounded, the following shows three possible ways, according to which we can classify previous methods:

$$Y = g(W, X) \overset{?}{=} h_1(WX) \tag{1}$$

$$\overset{?}{=} h_2(W)X \tag{2}$$

$$\overset{?}{=} W \cdot h_3(X). \tag{3}$$

NIA [3] assumes (1) and further that h_1 is an addition with some noise. Mask [4] assumes (2) and that h_2 is an elementwise multiplication. We take the same approach as [4] but makes no assumption on h_2, but instead identify it through machine learning. Equation (3) is another possibility, but does not seem very useful in modeling MVM distortion.

Later, in Section VI-C, we show through simulation that (2) is sound for IR drop-induced distortion. The soundness of (2) can be explained as follows. If we limit ourselves to the steady-state behavior during MVM computation, an RCA can be seen as a resistive network with constant, albeit programmable, resistance values. Then, despite the complex pathways in the crossbar, the output current should be linear to the input voltage, meaning that scaling and superposition properties on X should hold, hence, (2).

Thus, we only need to predict W_e from W. We call the former *effective weight* and the latter *programmed weight*. To recap, the mask method [4] uses the following simple function:

$$W_e = W \circ M \tag{4}$$

where $\circ$ is the Hadamard product (i.e., elementwise multiplication) and M has the same size as W. The intuition behind this method is that the physical location within an RCA has a dominant effect on the deviation of W_e from W. As we show in Section VI, this method has limited accuracy. This is because a mask can capture only the first-order effect (i.e., distance from input/output) but ignores second-order effects such as the resistance values of neighbor ReRAM cells, which is also important to get correct W_e.

C. Impact of IR Drop Problem

Ideally, the absolute value of the sensed current per a ReRAM cell should be constant, regardless of the position of the cell in the array. But due to the existence of wire resistance, which is inevitable in any interconnect, the sensed current can vary depending on the location. The variation in the sensed current is due to the voltage drops over the wire resistances that create leakage paths throughout the array which is referred as the sneak path problem.

To illustrate the effect of sneak paths, Fig. 3 plots the sensed current from each cell while simulating random binary weights. The sensed current is the *perceived* weight value as seen at the output port. The graphs clearly show that the sensed current decays exponentially across the diagonal direction of an array instead of having constant values in $\{-1, 1\}$. Also, with increasing wire resistance, the weights decay much faster.

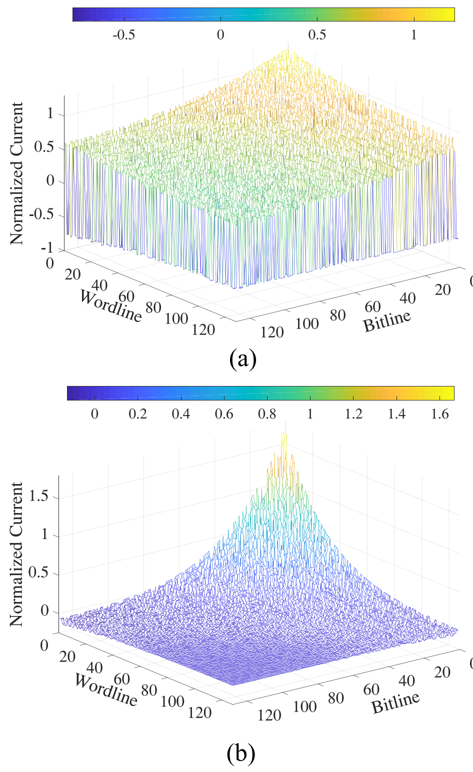


Fig. 3. Normalized sensed current for 128 × 128 crossbar array with one reference column. (a) $R_w = 0.1\ \Omega$. (b) $R_w = 2\ \Omega$.

D. DNN Application

The problem can be stated as follows: given a DNN, a train dataset, and the physical parameters of RCAs, such as LRS, HRS, wire resistance, array size, etc., to find the best synaptic weights for the DNN to be programmed into RCAs such that the accuracy of inference on RCAs can be maximized despite the existence of sneak path currents.

The DNN part of the problem can be solved easily using the backpropagation algorithm. The new challenge is how to quickly and accurately estimate the effect of sneak path currents during training so that we can guide the backpropagation algorithm to minimize or even compensate for the sneak paths. Note that sneak path patterns can vary widely and nonlinearly depending on the programmed binary weights, requiring re-evaluation of sneak paths for every weight update.[2] Also, since the number of RCAs in a DNN can be quite large, a naïve integration of SPICE simulation during training would require a prohibitive amount of resources.

Now, if we limit ourselves to the steady-state behavior, a passive RCA can be seen as a resistive network with constant, albeit programmable, resistance values. Then, we can use the result of previous work [21] on a generic resistive network model, which is shown to be as accurate as SPICE simulation. Importantly, this model implies that despite the presence of sneak path currents, a resistive network's output should be

[2]Possibly for every input as well if we do not take advantage of (5).

linear to input. Therefore, we can express the observed output current Y as the product of input voltage X and some matrix W_e

$$Y = g(W, X) = h(W)X = W_eX. \tag{5}$$

Here, W is the *programmed weight matrix* and W_e is the *effective weight matrix*. They have the same size as RCAs, which is $n \times m$. For QNN applications, input X is binary and weight matrix W is fixed point, whereas effective weight matrix W_e is real valued.

Hence, the goal is to find W_e from W. One may use SPICE simulation or equivalent numerical models such as [21], both of which are extremely slow. Inference, fortunately, does not require many SPICE simulations; one simulation per RCA is enough due to (5). This allows us to build an accurate and practical evaluation setup for sneak path-aware inference on RCAs. For training, however, we need a very fast model, and preferably in a closed-form expression so that it can be directly integrated into an existing DNN training framework. Also, for backpropagation to work, the model must be differentiable. This suggests that we can use regression methods to identify the function g, treating W and W_e as the input and output of the system to be identified.

In the next section, we present our solution to the regression problem. Note that though in this article we obtain W_e from SPICE or a SPICE-equivalent model, it could also be derived from real device measurement data, to which our technique should be applicable as well.

IV. Prediction of Effective Weight

While the problem of predicting W_e from W can be seen as a regression problem, typical regression functions such as linear regression cannot be directly applied due to the high-dimensionality of input/output data in our problem. With a 128×128 RCA, for instance, both input W and output W_e have 16K dimensions, requiring 256 million parameters for linear regression. We suggest two neural network models that can effectively predict W_e as follows.

A. Row-Column Network

Row-column network (RCN) is a nonlinear model, built by stacking multiple neural network layers. IR drop exists in both row and column directions, thus combining rowwise and columnwise models makes sense. We first define two layers and combine them to create RCN.

Parallel Linear Layer: We divide the input and output matrices into rows. Each row has its own linear regression parameters, followed by nonlinear activation function. In what follows, we use X and Y to refer to generic input and output matrices (of the same size as W):

$$\mathbf{PL_{row}} : X \rightarrow Y \text{ where } Y^{[i]} = f\left(X^{[i]} \cdot R_i + \vec{b}_i\right) \tag{6}$$

where $X^{[i]}$ is the ith row of X and f a nonlinear activation function (we use *tanh*), R_i a weight matrix of size $m \times m$, and $\vec{b}_i$ an m-dimensional bias vector. In neural network frameworks, this layer can be easily implemented as a hierarchy of primitive layers: a parallel layer and linear layers below it (hence, called *parallel linear layer*).

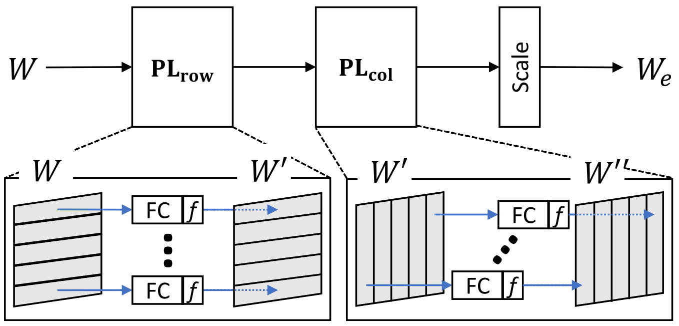


Fig. 4. RCN consisting of $\mathbf{PL_{row}}$, $\mathbf{PL_{col}}$, and an elementwise scale layer.

Columnwise parallel linear layer is defined similarly

$$\mathbf{PL_{col}}(X) = \mathbf{PL_{row}}\left(X^T\right)^T. \tag{7}$$

Then, RCN is defined to capture both row- and column-wise dependencies by stacking parallel linear layers of both directions

$$\mathbf{RCN}: \ W_e = \mathbf{PL_{col}}(\mathbf{PL_{row}}(W)) \circ U \tag{8}$$

where U is an $n \times m$ parameter matrix for the elementwise scale layer, which is also trainable. It adjusts the output range of RCN to match with W_e which can go beyond $[-1, 1]$. Fig. 4 illustrates parallel linear layers and RCN.

We train the network to minimize the mean squared error (MSE) loss, defined as

$$L = \frac{1}{N}\left\|\hat{W}_e - W_e\right\|_2^2 \tag{9}$$

where $N = nm$ is the number of elements in W_e, and $\hat{W}_e$ is the estimated effective weight matrix.

B. Scaling Convolutional Network

The scaling convolutional network (SCN) is our convolution approach to regression. Convolution layers are good at capturing spatial patterns in the input with a small number of parameters. On the other hand, SCN is clearly distinguished from CNNs for image classification; the output is not class labels, but a transformed version of the input, with the same size and data type, as the input.

To simplify the process of designing a new network, we leverage the mask idea to capture the spatial correlation, and add convolution layers to compensate for the effect of ReRAM cell values. At the top level, SCN consists of two elementwise scaling layers and a convolutional network in between as illustrated in Fig. 5.

The internal convolutional network is defined to be a stack of n convolutional layers, each with c output channels, except for the last (which has a single channel), where n and c are design parameters. The convolution filter size is fixed to 3×3, and padding and stride are both 1; no pooling layer is used. Activation function, ReLU, is used in all layers except the last one.

Specifically, the basic block CL_c is a convolutional layer with c output channels. We replicate the basic block $n - 1$

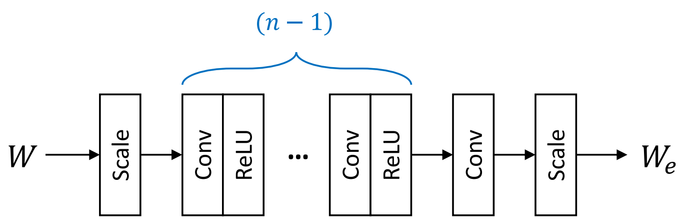


Fig. 5. SCN consisting of $n-1$ Conv-ReLU blocks, one convolution layer, and two elementwise scale layers.

times and add one convolutional layer having a single channel

$$\mathbf{CL}_c\colon X \to Y \text{ where } Y = \mathrm{ReLU}(\mathrm{Conv}_c(X)) \quad (10)$$

$$\mathbf{SCN}_{n,c}\colon W_e = \mathrm{Conv}_1(\mathbf{CL}_c(\cdots(\mathbf{CL}_c(W \circ U_1)))) \circ U_2. \quad (11)$$

The performance of SCN depends on the value of n and c. Through exploration using a randomly generated dataset for 128 $\times$ 128 array with wire resistance of 1 Ω (more details in Section VI-A), we have found that $\mathrm{SCN}_{7,32}$ gives the best balance between performance and model size; further increasing the hyperparameters did not give much performance improvement. SCN is also trained to minimize MSE.

Once the regression models are trained, they are integrated into the target DNN's training framework [see Fig 2(c)], so that the training of target DNNs can find the best *programmed weight* under *predicted* MVM distortion. During this training, the regression model's parameters are fixed.

V. Efficient Training of RCA-Mapped DNN Model

We now present our method for training a target neural network. A target neural network is a network whose inference is accelerated by the use of RCAs. In other words, a target DNN is a DNN model whose operations are mapped in part or in full to RCAs.

The key difference between training an RCA-mapped DNN model and training an ordinary DNN model is whether to consider RCA's nonideality such as output distortion due to IR drop, which we find can make training much harder to converge. Thus, in addition to a nonideality-aware training flow as has been suggested in previous work [4], we propose in this article a novel incremental training framework based on the concept of *nonideality modulation*. The idea is that by gradually increasing the degree of nonideality, we can achieve a much higher training performance, with a similar increase in the final validation accuracy as we demonstrate in our experimental results.

A. IR Drop-Aware Training

Using a prediction model from W to W_e, we train a target network (i.e., an RCA-mapped DNN model) as follows. First, we pretrain the baseline DNN model without considering RCA nonideality. This pretrained model is already quantized according to how the RCA-mapped DNN model is quantized, but no knowledge of RCA (such as how weight parameters are partitioned into RCAs) or its nonideality is used in the training. The pretrained model is used as the initial weight for IR drop-aware training. Second, for the forward propagation of IR drop-aware training, we modify weight matrices mapped to RCAs in this way: 1) weight matrices are partitioned into crossbar-sized submatrices; 2) the submatrices are transformed into new matrices of the same size to reflect the weight distortion due to nonideality by invoking a nonideality prediction model such as our SCN; and 3) the transformed submatrices are combined back to the original size. Third, we perform training of the target DNN using the modified weight matrices. The training itself is exactly the same as ordinary DNN training, but the difference is that we use the weight matrices modified in the previous step. Performing a simple MVM operation using the modified matrices can simulate the distorted MVM operation as in (5). In the backpropagation stage, the parameters of each layer are updated using the gradient of task loss (w.r.t. the parameters), and since the computation of the gradient involves the value of the modified weight matrices via the chain rule, the gradient descent algorithm will adjust weight W in a way to best minimize the task loss in the presence of IR drop. When calculating ΔW, we found that using straight through estimator (assuming $\Delta W = \Delta W_e$) is good enough to train the network. All the other parameters are propagated and updated following the exact chain rule. Fig. 6 describes how weights are transformed and updated during the DNN training. After training is done, we can validate the network with updated W.

Note that since our SCN network is a sequence of neural network layers, it can be easily integrated into a target neural network, and the evaluation of the integrated model can be done very efficiently on GPUs.

B. Incremental Training

While the IR drop-aware training procedure described above should be able to find the optimal weight parameters, we find that training often reach only much lower performance. This is likely due to gradient descent being stuck at one of local minima, and we find that getting out of local minima is not easy using common training recipes.

Our proposed method is based on the observation that often the degree of nonideality can be adjusted easily. For instance, the amount of distortion due to IR drop is proportional to the value of R_w. Thus, if our intended R_w is 2 for instance, we gradually increase R_w during training from 0.1 to 1 and to 2, which seems to help gradient descent-based training to better cope with changes in the search space, as compared with going from $R_w = 0$ to $R_w = 2$ at once.

It would be better if we can modulate nonideality during a single training session. While R_w is a real-valued parameter and we can certainly change its value continuously, the bottleneck is in the creation of an IR drop prediction model, which is specific to a certain R_w value. This is why we modulate R_w value in a discrete fashion. However, if one uses an IR drop prediction model that can take R_w as a continuous parameter, we can module R_w during training in a continuous fashion.

VI. Experiments

A. Experimental Setup

To evaluate the effectiveness of our proposed technique, we have extended the BinaryNet framework [22] for MNIST,

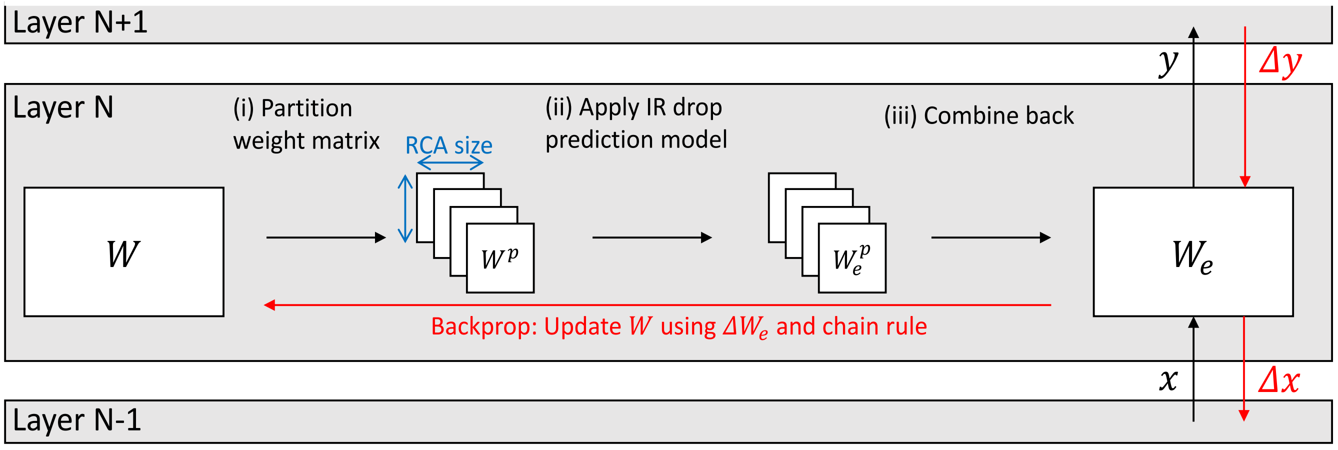

Layer N+1
Layer N
Layer N-1
(i) Partition weight matrix
RCA size
(ii) Apply IR drop prediction model
(iii) Combine back
W
W^p
W_e^p
W_e
y
Δy
x
Δx
Backprop: Update W using ΔW_e and chain rule

Fig. 6. IR drop-aware training flow. Red color indicates backpropagation stage.

TABLE I
NETWORK AND TRAINING PARAMETERS

	MNIST	CIFAR10	SVHN
Network type	MLP	CNN	CNN
#Layers	4	9	9
Initial training #epochs	100	500	200
Retraining #epochs	50	200	80
Baseline test accuracy (Binary)	98.41%	88.62%	97.18%
Baseline test accuracy (2-bit)	98.48%	89.09%	96.61%
Baseline test accuracy (8-bit)	98.52%	88.55%	96.66%

CIFAR10, and SVHN datasets.[3] Table I lists the key parameters of the networks and of training. We also use 2- or 8-bit weight quantized version of the networks.[4] Our primary metric is validation accuracy on unseen data (see Section III-A). The baseline accuracy is the test accuracy on GPU, which is the highest we can expect for validation accuracy. For the training under distorted MVM (step 3 of Fig. 2) we retrain the network from the baseline trained weight, which is commonly called *retraining*. For retraining, we reduce the initial learning rate to 1/8 as that of the baseline training. Incremental training (Section V-B) is applied to cases where retraining accuracy is low, which are all the SVHN cases with $R_w > 0.1$.

We compare the following five cases: NIA, mask, RCN, SCN, and the validation accuracy from step 2 (referred to as "w/o retraining"). We evaluate two different crossbar sizes, 64×64, and 128×128. We use the following device parameters, taken from recently fabricated devices [23]: LRS $=$ 1E3 Ω, HRS $=$ 1E6 Ω. The wire resistance per cell (R_w) is varied from 0.1 to 2 Ω. This range agrees with that of many previous work [21], [24], [25]. Li *et al.* [24] matched the experimental setup with the simulations and the wire resistance is found to be about 0.32 Ω/block (1T1R) with 2-μm feature size for transistors, and [25] uses 1 Ω/block (1T1R). Fouda *et al.* [21] reported that the estimated wire resistance is 1.908 Ω/segment (0T1R) for 50-nm feature size. Hence, we wanted to test different scenario within these technologies. It is also worth mentioning that lower technologies nodes would experience an exponential increase in the wire resistance which might not be practical to fabricate functioning ReRAM. These parameters are also very similar to what is used in other previous work [3], [4], including the device-to-wire resistance ratio. We consider the interconnect wire resistance, which is the main cause of the IR drop problem. Other nonidealities such as driver and load resistances are not included, since their effects can be eliminated by designing better circuits and therefore considered not essential.

To train the IR drop prediction models, we use 50 000 randomly generated crossbar-sized data as weight (W) and corresponding effective weights (W_e) generated from SPICE simulations for each scenario. We store random data in crossbar arrays and we measure the effective conductance value by measure the output current. The random data help to get an average model that could work for any application, regardless of spatial or repetition of the weights. For NIA, we use the trained weights of each target network as described in [3]; thus, the number of data samples differs depending on the network and the crossbar size. For the mask method, we follow the procedure in [4], which uses 100 random data samples. Note that we train the IR drop prediction models again for each device parameter combination (RCA size and wire resistance), but use the same trained model across different networks and between training versus test. The only exception is the NIA method, which is trained again for each network as well as for each device parameter combination.

B. Neural Network Accelerator Setup

In this article, we evaluate networks with binary ($\{-1, +1\}$), 2-bit (five states), and 8-bit (257 states) weights. We assume multistate ReRAM cells such that one weight parameter can be represented by one ReRAM cell.

Due to the positivity of ReRAM conductance, a special mechanism is necessary in order to represent negative weight values. We store and represent weight values using a reference column, as is done in [4]. A reference column is an extra column in an RCA, which is shared with all the other columns in the RCA and has the following conductance value: $G_r = (G_{\max} + G_{\min})/2 \approx G_{\max}/2$. A ReRAM cell with the conductance value of G ($G_{\min} \le G \le G_{\max}$) can represent the

[3]https://github.com/itayhubara/BinaryNet

[4]Our n-bit weight quantization uses $2^n + 1$ states due to the conductance mapping scheme in Section VI-B.

TABLE II
WEIGHT-TO-CONDUCTANCE MAPPING EXAMPLES

Weight	Binary	2-bit
1	$G_{\max}$	$G_{\max}$
0.5	–	$\frac{3G_{\max}}{4}$
0	–	$\frac{G_{\max}}{2}$
-0.5	–	$\frac{G_{\max}}{4}$
-1	$G_{\min}$	$G_{\min}$

weight value of $(G - G_r)/\Delta$, where Δ is a scale factor set to $G_{\max}/2$, so that the range of values represented by a ReRAM cell corresponds to $[-1,\ 1]$, which is the range of quantized weight values. Table II shows binary and 2-bit (five states) examples. Binary has only two values, which are assigned to $G_{\max}$ and $G_{\min}$. The 2-bit case has five values in steps of 0.5, which corresponds to $(G_{\max} - G_{\min})/4 \approx G_{\max}/4$ in conductance.

Note that this weight-to-conductance mapping is more efficient in terms of the number of RCAs than two-device realization (i.e., using two ReRAM cells per weight parameter). We assume that the subtraction operation needed to implement the weight realization is implemented as digital circuit after current-to-voltage conversion and ADC (analog-digital conversion).

In addition, we assume that convolution layers are mapped to RCAs by parallelizing the input channels and output channels [26]. That is, for an $N \times N$ RCA, an N-dimensional vector corresponding to N input channels is given to an RCA as input voltage, and the RCA generates as output another vector corresponding to N output channels. This is repeated so that all MAC operations of a convolution can be covered.

C. Validation of Linearity Principle

We show through simulation that (2) is sound for IR drop-induced distortion. To show, we run IR drop simulation [3] using trained weights of an MNIST BNN as W. The network is identical to the one used in Section VI except that it has 1024 hidden neurons in each hidden layer. Note that the network's weights are spread across 736 crossbar tiles of 64×64, meaning that there are 736 different weight matrices to use for this experiment. For X, we use input activations during MNIST inference (for the first 500 test images). We record Y vectors, which are output current from crossbar arrays obtained via IR drop simulation.

From the collected X, Y pairs, we set up a linear equation, $W_e X = Y$, one per each crossbar, which is underparameterized and can be solved for W_e to minimize the MSE, $\mathcal{E} = \text{mse}(W_e X - Y)$. We report the error using the W_e found. If (2) is sound, the error should be very small. Indeed the error turns out to be within rounding error (2.09E-11), confirming that (2) is sound.

D. MVM Computation Accuracy

We first evaluate the accuracy of various prediction methods,[5] that is, how close their prediction is to SPICE simulation result. For this experiment, we use 1000 random weight matrices quantized to binary, 2- and 8-bit as well as 1000 random binary input vectors.

[5]NIA and mask in the previous work are not intended to be a prediction method, but they can be viewed as one.

TABLE III
MSE COMPARISON (RCA SIZE: 128×128, $R_w = 1\Omega$)

	Binary	2-bit	4-bit	8-bit
NIA	3.21E+0	2.18E+0	1.94E+0	1.85E+0
Mask	1.13E+0	1.14E+0	1.17E+0	1.28E+0
RCN	2.61E-2	1.69E-2	1.47E-2	1.36E-2
SCN	1.51E-2	5.27E-3	3.40E-3	2.77E-3

The results summarized in Table III are carried out with $R_w = 1\ \Omega$ and 128×128 RCA. We compare the MSE of MVM output (RCA output current), averaged over weight matrices. The table clearly shows that our prediction models are definitely more accurate in predicting RCA behaviors than the previous methods in terms of MSE by about 2–3 orders of magnitude. Though the low MSE is not enough to guarantee high performance in the DNNs, our methods also work for large networks. We examine the DNN validation results in the next section.

TABLE IV
BNN VALIDATION ACCURACY (%)

RCA		64x64			128x128		
$R_w(\Omega)$		0.1	1	2	0.1	1	2
MNIST	None	97.63	20.20	11.55	73.01	9.94	9.90
	NIA	98.41	96.27	80.03	98.13	28.15	12.73
	Mask	98.44	96.85	86.27	97.94	37.10	21.73
	RCN	**98.45**	**98.28**	97.92	**98.31**	**97.80**	85.47
	SCN	98.42	98.20	**98.13**	98.22	97.71	**96.92**
CIFAR10	None	46.60	10.00	10.00	10.00	10.00	10.00
	NIA	55.34	10.00	10.00	17.25	10.00	10.00
	Mask	86.17	10.00	10.00	8.87	10.00	10.00
	RCN	87.24	78.30	44.60	82.40	65.07	38.19
	SCN	**87.34**	**84.29**	**76.92**	**83.87**	**82.22**	**71.97**
SVHN	None	94.22	9.16	7.76	9.16	19.59	19.59
	NIA	93.37	6.70	6.70	57.04	6.70	6.70
	Mask	95.50	6.35	19.59	12.43	7.59	19.59
	RCN	96.38	95.59	62.95	**96.78**	92.48	25.94
	SCN	**96.39**	**96.27**	**95.45**	96.50	**96.48**	**94.92**

E. Network Training Performance

1) BNN Results: Table IV shows the validation accuracy for MNIST, Cifar10, and SVHN BNNs. *None* means the case without retraining. Again these results are from step 4, using unseen test data.

From the result, we make the following observations. First, the result confirms the severity of the IR drop problem in RCA-based neural networks. In particular, even when wire resistance is as low as 0.1 Ω, 128 × 128 RCA's MVM computation is so distorted that without retraining it is unusable even for MNIST BNN. Second, all retraining methods, including NIA and mask do help train the BNN. For instance, in the case of 64 × 64 RCA with 1 Ω, the validation accuracy without retraining is only about 20%, but through retraining, all methods recover the baseline-level accuracy. Third, our proposed methods consistently outperform the previous methods. The performance difference is more remarkable in the case of CIFAR10 and SVHN.

TABLE V
QNN (2-Bit) Validation Accuracy (%)

RCA		64x64			128x128		
$R_w(\Omega)$		0.1	1	2	0.1	1	2
CIFAR10	None	76.92	10.00	10.00	10.54	10.00	10.00
	NIA	87.10	11.38	10.00	30.05	10.00	10.00
	Mask	87.39	10.00	10.00	65.80	10.00	10.00
	SCN	**88.39**	**87.20**	**82.38**	**88.01**	**65.04**	**51.72**
SVHN	None	74.02	6.13	6.13	6.13	6.13	6.13
	NIA	80.81	6.40	6.13	6.13	15.94	9.69
	Mask	76.73	9.69	7.76	6.13	7.76	7.76
	SCN	**96.67**	**93.89**	**90.18**	**95.91**	**64.69**	**30.32**

TABLE VI
QNN (8-Bit) Validation Accuracy (%)

RCA		64x64			128x128		
$R_w(\Omega)$		0.1	1	2	0.1	1	2
CIFAR10	None	64.30	10.00	10.00	10.00	10.00	10.00
	NIA	88.41	9.87	10.00	53.03	10.00	10.00
	Mask	74.85	24.30	15.94	44.05	12.60	10.00
	SCN	**88.92**	**87.35**	**77.27**	**88.63**	**52.79**	**38.27**
SVHN	None	71.72	6.13	6.13	6.13	6.13	6.38
	NIA	81.62	6.12	19.59	6.13	15.94	**15.94**
	Mask	83.48	9.69	9.69	6.81	9.69	9.69
	SCN	**96.46**	**94.30**	**88.37**	**95.69**	**43.92**	14.76

The CIFAR10 and SVHN results unequivocally show the superiority of our proposed methods over the previous methods. In particular, the SCN model not only outperforms all the other methods but it also gives acceptable accuracy where the previous methods have completely failed such as 64x64 $R_W = 1$ case. Note that though the accuracy of "w/o retraining" for SVHN is sometimes near 20%, it is due to the unbalanced label distribution; for SVHN, 20% accuracy is no better than 10% or less.

2) QNN Results: To evaluate the scalability of our method, we perform the same set of experiments using multibit neural networks, called QNN. For the QNN model, we use the same network models as BNNs but with 2- or 8-bit weights. Input and output activations are kept binary, not only because doing so simplifies the hardware around RCAs but also because our focus in this work is in modeling the IR drop problem in weight realization. In this experiment we omit RCN in favor of SCN because SCN is shown to outperform RCN.

The results are summarized in Tables V and VI. Again, these results are validation accuracy obtained through SPICE-equivalent simulation, rather than retraining accuracy. We note that retraining accuracy is very similar to the baseline accuracy in all the cases, indicating that training itself has been successful except a few cases for SCN (see Table VII). Compared with the BNN results (see Table IV), we observe much higher accuracy drop with all methods, with a few exceptions. Note that there is no significant difference in the baseline accuracy, i.e., test accuracy on GPU, among the BNN and QNNs (see Table I).

First, the universal drop in validation accuracy indicates that the problem of predicting IR drop patterns becomes harder as weight precision increases. This is understandable, since conductance values in, say, 8-bit RCAs are more densely populated than those of 2-bit RCAs (see Table II), making them more susceptible to distortion. Second, our SCN method shows consistently higher performance than all the previous methods considered, often with a large margin, which demonstrates the effectiveness of our method in dealing with IR drop on RCAs.

Third, with SCN, not all cases are worse at higher precision. The best example is the CIFAR10 with 64 × 64 RCAs, for which 2 and 8-bit QNNs show higher performance than with BNN (for all R_w values). The 128 × 128 case with $R_w = 0.1$ exhibits a similar trend, though no such exceptions are seen in the SVHN result. One way to understand this result is to see IR drop mitigation as a Boolean outcome (success/fail). The important parameter here is not the validation accuracy when it succeed, which is anyway the result of a stochastic process, but when it starts to fail or the difficulty level, which is determined by R_w, RCA size and precision. Thus, the effect of higher precision is to make SCN ineffective at a lower R_w value.

In summary, QNNs represent a more challenging case for the IR drop prediction problem, but our SCN method, which vastly outperforms the previous methods, can achieve a similar level of network accuracy recovery as with BNN, although the range of cases where SCN succeeds is narrower at higher precision.

TABLE VII
SCN Retraining Accuracy (%)

RCA		64x64			128x128		
$R_w(\Omega)$		0.1	1	2	0.1	1	2
Binary	MNIST	98.49	98.49	98.40	98.36	98.20	97.77
	CIFAR10	86.26	85.12	83.94	84.09	82.69	78.27
	SVHN	96.63	96.59	95.81	96.98	96.56	95.59
2-bit	CIFAR10	88.67	88.27	83.41	88.97	84.37	79.51
	SVHN	96.59	95.81	94.10	96.35	85.49	77.18
8-bit	CIFAR10	88.72	87.84	82.71	89.18	85.84	80.10
	SVHN	96.59	95.71	94.41	96.54	88.26	77.74

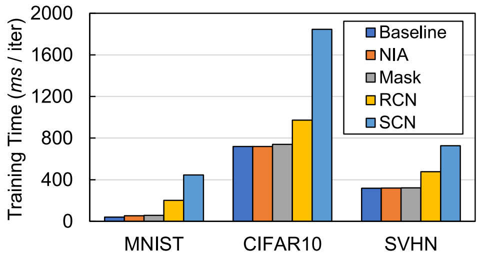


Fig. 7. Training time comparison (per iteration).

F. Time Overhead Comparison

Fig. 7 compares the training time of BNNs using various IR drop mitigation methods. The BNN framework is implemented in Torch7, and training is done using a GPU (Nvidia GeForce GTX 1080Ti) on a system with Intel Xeon CPU E5-2630 v4. The total training time depends on the number of iterations and the number of epochs as well. Though the RCN and SCN cases take longer to train, which is due to the evaluation of an additional neural network within the BNN training, the increased training time is justifiable, given the significant improvement in accuracy as shown in Table IV.

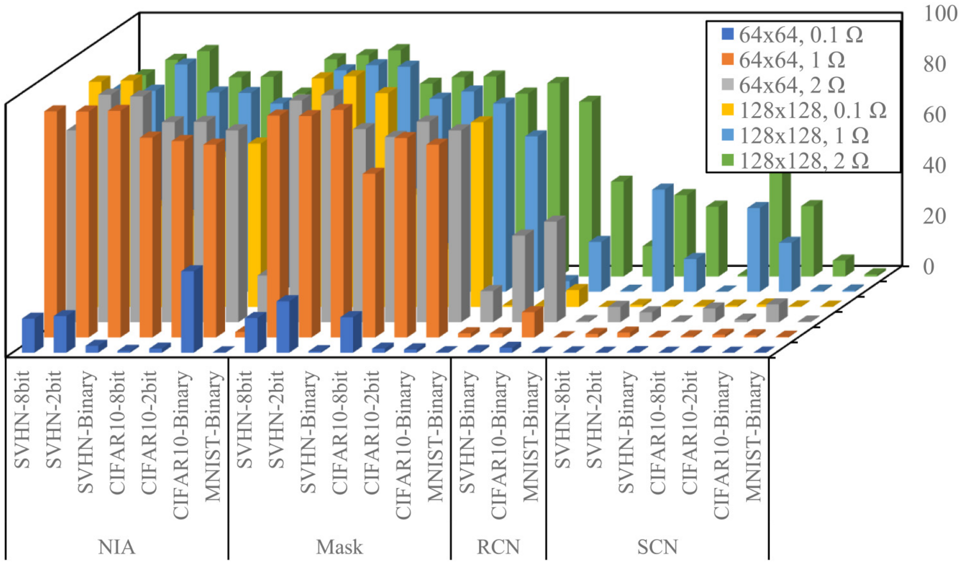


Fig. 8. Accuracy drop (pp) of retrained networks compared to retraining accuracy.

Also, compared to other IR drop estimation methods such as SPICE simulation, the overhead of RCN and SCN is negligible.

G. Accuracy Drop From Retraining Accuracy

The SCN and other methods we consider in this article can be regarded as IR drop prediction methods. From that perspective, the retraining accuracy, which is the test accuracy on GPU after IR drop-aware retraining, should mirror the final, validation accuracy as closely as possible. The higher the gap between retraining and validation accuracy is, the less useful the retraining accuracy will be in terms of predicting the accuracy on real ReRAM hardware.

Thus, we compare different IR drop prediction methods in terms of retraining–validation accuracy gap for various cases. The results are summarized in Fig. 8, where the lower the accuracy gap is, the more accurate and reliable the method is in predicting the effect of IR drop. The results suggest that all the methods are quite accurate for the easiest case (64×64, $R_w = 0.1\ \Omega$), but as the cases become more challenging, NIA and Mask quickly become highly inaccurate. Our SCN method, on the other hand, can reduce the gap greatly compared with the other methods, achieving near zero gap for many cases. This result unequivocally demonstrates the superiority of the SCN method in predicting the IR drop effect on RCAs.

H. Effect of Incremental Training

To see the effect of incremental training, we perform identical training experiments with and without incremental training for SVHN 128×128 cases that show low retraining performance. For nonincremental training, we use the pretrained weight ($R_w = 0$) as the initial weight and perform IR drop-aware training for a given R_w value of 1 or 2. In the case of incremental training, the $R_w = 1$ case is retrained using $R_w = 0.1$ training result as the initial weight, and the $R_w = 2$ case is trained using $R_w = 1$ training result as the initial weight.

TABLE VIII
SCN Incremental Training/Validation Accuracy (%)

SVHN 128x128		Non-incremental		Incremental	
$R_w(\Omega)$		1	2	1	2
Training	Binary	95.63	94.48	96.56	95.59
	2-bit	80.37	36.65	85.49	77.18
	8-bit	81.93	38.44	88.26	77.74
Validation	Binary	95.62	89.37	96.48	94.92
	2-bit	64.69	30.32	72.57	49.63
	8-bit	43.92	14.76	47.95	45.61

Table VIII compares the results with and without incremental training. The table clearly shows that our incremental training consistently outperforms nonincremental training, not only in terms of retaining accuracy but even in terms of validation accuracy. Moreover the advantage of incremental training is often very significant. For instance, with 8-bit weight and $R_w = 2\ \Omega$, incremental training achieves more then twice the retraining accuracy as well as over 30-pp improvement in validation accuracy compared with nonincremental training. This result demonstrates that our IR drop-aware training coupled with incremental training can find weight parameters of a target DNN that can effectively mitigate the nonideality due to IR drop in RCAs.

I. Effect of Device Variation

The ReRAM device's variability could cause nonnegligible drop in the DNN accuracy [27], [28]. Hence, we evaluate the impact of device variation under the IR drop problem. During validation, we inject Gaussian noise on W_e, which we get from crossbar simulation during the validation stage. The noise-injected weight W_n with noise parameter ρ follows:

$$W_n = W_e + \mathcal{N}(0, \sigma_{W_e}^2) = W_e(1 + \mathcal{N}(0, \rho^2)) \qquad (12)$$

where $\sigma_{W_e} = \rho \cdot W_e$.

TABLE IX
SCN VALIDATION ACCURACY DROP (PP) WITH NOISE INJECTED TO WEIGHT VALUES (64×64, $R_w = 1\ \Omega$)

	ρ (%)	w/o retraining 2-bit	w/o retraining 8-bit	w/ retraining 2-bit	w/ retraining 8-bit
CIFAR10	5	1.66	1.88	0.03	0.10
	10	5.45	6.69	1.45	2.84
	20	19.17	23.90	10.41	15.13
SVHN	5	1.26	1.62	0.34	0.04
	10	5.73	7.18	1.50	1.90
	20	39.53	41.94	11.16	13.05

Table IX shows the accuracy drop of SCN under noise for some cases. (See Tables V and VI for performance without noise) To compensate the effect of noise, we fine-tune the network with noise-injected effective weight. For $\rho = 5$, the effect of noise is relatively small and almost no loss is after retraining. For higher noise parameters the effect becomes more significant. In SVHN $\rho = 20$ case, for example, it suffers over 40 pp of accuracy drop but it is compensated around 13 pp after fine-tuning. These results show that our SCN method can be robust to weight variation if we are aware of noise when training.

VII. CONCLUSION

We presented a novel method to incorporate the IR drop problem during BNN and Q NN training with a negligible overhead. Compared to hardware methods (e.g., new device/selector material, error compensating circuitry), our training method is essentially free, and applicable on top of any hardware methods. Our experimental results demonstrate that while the IR drop problem renders many passive ReRAM crossbar configurations unsuitable for DNN inference, our proposed method can extend the range of usable configurations significantly, achieving near-baseline level test validation accuracy with MNIST and SVHN BNNs, and a significant boost with CIFAR10 BNN and QNNs.

We see many paths for future work. ReRAMs have many nonidealities including variability, stochastic noise, and permanent faults, with some of them very damaging. Training in the presence of unpredictable nonidealities is left for future work.

REFERENCES

[1] S. Lee, G. Jung, M. Fouda, J. Lee, A. Eltawil, and F. Kurdahi, "Learning to predict IR drop with effective training for ReRAM-based neural network hardware," in *Proc.57th Annu. ACM/IEEE Design Autom. Conf. (DAC)*, Jul. 2020, pp. 1–6.

[2] Q. Xia and J. J. Yang, "Memristive crossbar arrays for brain-inspired computing," *Nat. Mater.*, vol. 18, no. 4, pp. 309–323, 2019.

[3] Z. He, J. Lin, R. Ewetz, J.-S. Yuan, and D. Fan, "Noise injection adaption: End-to-end ReRAM crossbar non-ideal effect adaption for neural network mapping," in *Proc. 56th Annu. Design Autom. Conf.*, 2019, p. 57.

[4] M. E. Fouda, S. Lee, J. Lee, A. Eltawil, and F. Kurdahi, "Mask technique for fast and efficient training of binary resistive crossbar arrays," *IEEE Trans. Nanotechnol.*, vol. 18, no. 1, pp. 704–716, Jan. 2019.

[5] A. Shafiee *et al.*, "ISAAC: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars," *ACM SIGARCH Comput. Archit. News*, vol. 44, no. 3, pp. 14–26, 2016.

[6] K. Chellapilla, S. Puri, and P. Simard, "High performance convolutional neural networks for document processing," in *Proc. 10th Int. Workshop Front. Handwriting Recognit.*, Oct. 2006, p. 1–6.

[7] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. 32nd Int. Conf. Int. Conf. Mach. Learn.*, vol. 37, 2015, pp. 448–456.

[8] H. Yonekawa and H. Nakahara, "On-chip memory based binarized convolutional deep neural network applying batch normalization free technique on an FPGA," in *Proc. IEEE IPDPS Workshops*, May 2017, pp. 98–105.

[9] P. Chi *et al.*, "PRIME: A novel processing-in-memory architecture for neural network computation in ReRAM-based main memory," in *Proc. 43rd Int. Symp. Comput. Archit.*, Piscataway, NJ, USA, 2016, pp. 27–39.

[10] L. Song, X. Qian, H. Li, and Y. Chen, "PipeLayer: A pipelined ReRAM-based accelerator for deep learning," in *Proc. IEEE Int. Symp. High Perform. Comput. Archit. (HPCA)*, Feb. 2017, pp. 541–552.

[11] Y. Cai *et al.*, "Long live TIME: Improving lifetime for training-in-memory engines by structured gradient sparsification," in *Proc. 55th Annu. Design Autom. Conf.*, 2018, pp. 1–6.

[12] X. Qiao, X. Cao, H. Yang, L. Song, and H. Li, "Atomlayer: A universal reRAM-based CNN accelerator with atomic layer computation," in *Proc. 55th Annu. Design Autom. Conf.*, 2018, pp. 1–6.

[13] C.-T. Ho and A. B. Kahng, "IncPIRD: Fast learning-based prediction of incremental IR drop," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, 2019, pp. 1–8.

[14] C.-H. Pao, A.-Y. Su, and Y.-M. Lee, "XGBIR: An XGBoost-based IR drop predictor for power delivery network," in *Proc. Design Autom. Test Europe Conf. Exhibit. (DATE)*, 2020, pp. 1307–1310.

[15] Z. Xie *et al.*, "PowerNet: Transferable dynamic IR drop estimation via maximum convolutional neural network," in *Proc. 25th Asia South Pacific Design Autom. Conf. (ASP-DAC)*, 2020, pp. 13–18.

[16] Z. Xie, H. Li, X. Xu, J. Hu, and Y. Chen, "Fast IR drop estimation with machine learning: Invited paper," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, 2020, pp. 1–8.

[17] C. Huang, N. Xu, K. Qiu, Y. Zhu, D. Ma, and L. Fang, "Efficient and optimized methods for alleviating the impacts of IR-drop and fault in RRAM based neural computing systems," *IEEE J. Electron Devices Soc.*, vol. 9, pp. 645–652, 2021.

[18] Y. Zhu, X. Zhao, and K. Qiu, "Insights and optimizations on IR-drop induced sneak-path for RRAM crossbar-based convolutions," in *Proc. 25th Asia South Pacific Design Autom. Conf. (ASP-DAC)*, 2020, pp. 506–511. [Online]. Available: https://doi.org/10.1109/ASP-DAC47756.2020.9045671

[19] S. Roy, S. Sridharan, S. Jain, and A. Raghunathan, "TxSim: Modeling training of deep neural networks on resistive crossbar systems," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 29, no. 4, pp. 730–738, Apr. 2021.

[20] I. Chakraborty, M. F. Ali, D. E. Kim, A. Ankit, and K. Roy, "GENIEx: A generalized approach to emulating non-ideality in memristive Xbars using neural networks," 2020, *arXiv:2003.06902*.

[21] M. E. Fouda, A. M. Eltawil, and F. Kurdahi, "Modeling and analysis of passive switching crossbar arrays," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 65, no. 1, pp. 270–282, Jan. 2018.

[22] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks," in *Proc. 30th Int. Conf. Neural Inf. Process. Syst.*, 2016, pp. 4114–4122.

[23] M. Hu *et al.*, "Memristor-based analog computation and neural network classification with a dot product engine," *Adv. Mater.*, vol. 30, no. 9, 2018, Art. no. 1705914.

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

[25] F. Zhang and M. Hu, "Memristor-based deep convolution neural network: A case study," 2018, *arXiv:1810.02225*.

[26] C. Zhang, P. Li, G. Sun, Y. Guan, B. Xiao, and J. Cong, "Optimizing FPGA-based accelerator design for deep convolutional neural networks," in *Proc. ACM/SIGDA Int. Symp. Field-Programmable Gate Arrays*, 2015, pp. 161–170. [Online]. Available: http://doi.acm.org/10.1145/2684746.2689060

[27] G. Charan, A. Mohanty, X. Du, G. Krishnan, R. V. Joshi, and Y. Cao, "Accurate inference with inaccurate RRAM devices: A joint algorithm-design solution," *IEEE J. Explor. Solid-State Computat. Devices Circuits*, vol. 6, no. 1, pp. 27–35, Jun. 2020.

[28] M. E. Fouda, S. Lee, J. Lee, G. H. Kim, F. Kurdahi, and A. Eltawil, "IR-QNN framework: An IR drop-aware offline training of quantized crossbar arrays," *IEEE Access*, vol. 8, pp. 228392–228408, 2020.

Sugil Lee received the B.S. degree in mathematical science from the Korea Advanced Institute of Science and Technology, Deajeon, South Korea, in 2017, and the M.S. degree in computer science and engineering from the Ulsan National Institute of Science and Technology, Ulsan, South Korea, in 2019, where he is currently pursuing the Ph.D. degree with the Department of Electrical Engineering.

His current research interests include HW-aware deep neural network training and their energy-efficient implementation methodologies.

Mohammed E. Fouda received the B.Sc. degree (Hons.) in electronics and communications engineering and the M.Sc. degree in engineering mathematics from the Faculty of Engineering, Cairo University, Cairo, Egypt, in 2011 and 2014, respectively, and the Ph.D. degree from the University of California at Irvine, Irvine, CA, USA, in 2020.

He currently works as a Researcher with the University of California at Irvine. He has published more than 130 peer-reviewed journal and conference papers, one Springer book, and three book chapters. His H-index is 23, and he has been cited more than 2000 times. His research interests include analog AI hardware, neuromorphic circuits and systems, brain-inspired computing, memristive circuit theory, fractional circuits and systems, and analog circuits.

Dr. Fouda was the recipient of the Best Paper Award in ICM for 2013 and 2020 and the Broadcom Foundation Fellowship for 2016/2017. He serves as a peer reviewer for many prestigious journals and conferences. He also serves as an Associate Editor for the *Frontiers in Electronics* and the *International Journal of Circuit Theory and Applications*, in addition to serving as a technical program committee member for many conferences.

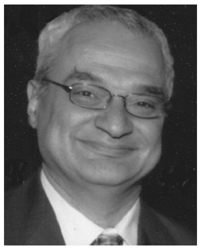
Jongeun Lee (Member, IEEE) received the B.S. and M.S. degrees in electrical engineering, and the Ph.D. degree in electrical engineering and computer science from Seoul National University, Seoul, South Korea, in 1997, 1999, and 2004, respectively.

Since 2009, he has been with the faculty of Ulsan National Institute of Science and Technology, Ulsan, South Korea, where he is a Professor of Electrical Engineering. His research interests include neural network processors, reconfigurable architectures, compilers, and hardware–software codesign.

Ahmed M. Eltawil (Senior Member, IEEE) received the B.Sc. and M.Sc. degrees (Hons.) from Cairo University, Giza, Egypt, in 1999 and 1997, respectively, and the Doctorate degree from the University of California at Los Angeles, Los Angeles, CA, USA, in 2003.

He is a Professor of Electrical and Computer Engineering with King Abdullah University of Science and Technology (KAUST), Thuwal, Saudi Arabia, where he joined the Computer, Electrical and Mathematical Science and Engineering Division in 2019. Prior to that, he was with the Electrical Engineering and Computer Science Department, University of California at Irvine, Irvine, CA, USA, in 2005. He is the Founder and the Director of the Communication and Computing Systems Laboratory, KAUST. His current research interests are in the general area of smart and connected systems with an emphasis on mobile systems.

Dr. Eltawil received several awards, including the NSF CAREER Grant supporting his research in low-power computing and communication systems. In 2021, he was selected as the Innovator of the Year by the Henry Samueli School of Engineering, the University of California at Irvine, where he received United States Congress Certificate of recognition for his contributions. He has been with the technical program committees and steering committees for numerous workshops, symposia, and conferences in the areas of low-power computing and wireless communication system design. He is a Senior Member of the National Academy of Inventors, USA.



Fadi Kurdahi (Fellow, IEEE) received the B.E. degree in electrical engineering from the American University of Beirut, Beirut, Lebanon, in 1981, and the Ph.D. degree from the University of Southern California, Los Angeles, CA, USA, in 1987.

Since 1987, he has been a Faculty Member with the Department of Electrical Engineering and Computer Science, University of California at Irvine, Irvine, CA, USA, where he conducts research in the areas of computer-aided design, high-level synthesis, and design methodology of large-scale systems, and serves as the Director of the Center for Embedded and Cyber-Physical Systems, comprised of world-class researchers in the general area of embedded and cyber–physical systems.

Dr. Kurdahi received the Best Paper Award of the IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS in 2002, the Best Paper Award at ISQED in 2006, four other Distinguished Paper Awards at DAC, EuroDAC, ASP-DAC, and ISQED, and also the Distinguished Alumnus Award from his Alma Mater, the American University of Beirut, in 2008. He was the program chair or the general chair on program committees of several workshops, symposia, and conferences in the area of CAD, VLSI, and system design. He has served on numerous editorial boards. He is a fellow of the AAAS.