Improving the accuracy and robustness of RRAM-based in-memory computing against RRAM hardware noise and adversarial attacks

IOP Publishing Semiconductor Science and Technology
Semicond. Sci. Technol. **37** (2022) 034001 (11pp) https://doi.org/10.1088/1361-6641/ac461f

Improving the accuracy and robustness of RRAM-based in-memory computing against RRAM hardware noise and adversarial attacks

Sai Kiran Cherupally[1], Jian Meng[1], Adnan Siraj Rakin[1], Shihui Yin[1], Injune Yeo[1], Shimeng Yu[2], Deliang Fan[1] and Jae-Sun Seo[1,*]

[1] Arizona State University, Tempe, AZ, United States of America
[2] Georgia Institute of Technology, Atlanta, GA, United States of America

E-mail: jaesun.seo@asu.edu

Received 8 June 2021, revised 25 October 2021
Accepted for publication 23 December 2021
Published 13 January 2022

Abstract

We present a novel deep neural network (DNN) training scheme and resistive RAM (RRAM) in-memory computing (IMC) hardware evaluation towards achieving high accuracy against RRAM device/array variations and enhanced robustness against adversarial input attacks. We present improved IMC inference accuracy results evaluated on state-of-the-art DNNs including ResNet-18, AlexNet, and VGG with binary, 2-bit, and 4-bit activation/weight precision for the CIFAR-10 dataset. These DNNs are evaluated with measured noise data obtained from three different RRAM-based IMC prototype chips. Across these various DNNs and IMC chip measurements, we show that our proposed hardware noise-aware DNN training consistently improves DNN inference accuracy for actual IMC hardware, up to 8% accuracy improvement for the CIFAR-10 dataset. We also analyze the impact of our proposed noise injection scheme on the adversarial robustness of ResNet-18 DNNs with 1-bit, 2-bit, and 4-bit activation/weight precision. Our results show up to 6% improvement in the robustness to black-box adversarial input attacks.

Keywords: IMC noise-aware training, RRAM-friendly DNNs, Adversarial defense with RRAM noise

(Some figures may appear in colour only in the online journal)

1. Introduction

Deep neural networks (DNNs) have demonstrated high accuracy in many computer vision and speech recognition tasks, but they require large amounts of storage and lead to high latency and energy consumption. Many recent works have been presented to address such issues. At the software level, works such as [1–3] addressed the issue of storage by aggressively reducing the DNN precision, with minimal accuracy degradation. On the hardware side, many digital accelerators have been proposed that address these issues by using optimized dataflows, systolic arrays of multiply-accumulate (MAC) units, and memory hierarchies [4–6]. However, most of these digital designs have a physical separation between the computation unit and memory storage, which makes the power consumption and latency of the DNN hardware to be dominated by memory accesses and data communication (almost up to two-thirds).

To address such bottlenecks, the in-memory computing (IMC) paradigm that integrates MAC computation inside the memory itself emerged as an effective solution. The

* Author to whom any correspondence should be addressed.

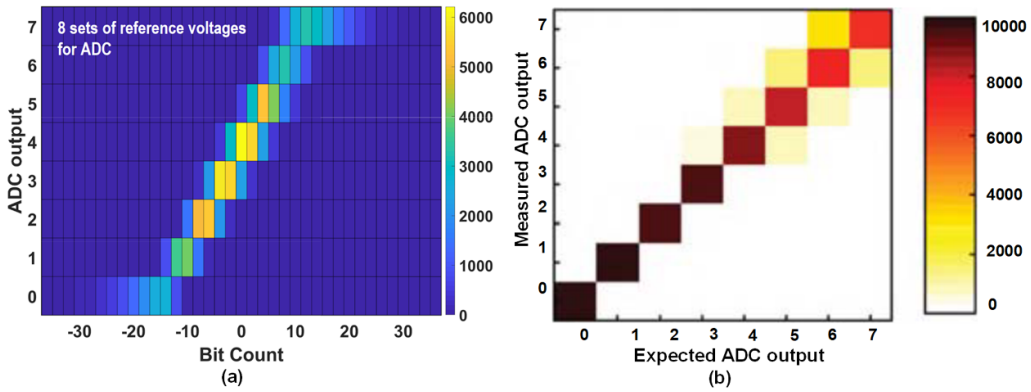

Figure 1. Analog in-memory computing involves inherent variability and noise, as shown in the prototype chip measurements: (a) © [2019] IEEE. Reprinted, with permission, from [7]. (b) © [2021] IEEE. Reprinted, with permission, from [8].

IMC designs using SRAM or emerging non-volatile memory (NVM) activate multiple/all rows of the array simultaneously and generate the MAC values in parallel in the analog domain. Then, an analog-to-digital converter (ADC) at the periphery digitizes the MAC value. Since MAC operation is a core computation in DNNs, IMC can significantly improve the energy consumption of DNN hardware by minimizing memory accesses. Recent IMC prototype chips have reported high energy efficiency [7, 9–12].

Although the IMC technique significantly improves the energy-efficiency compared to other digital accelerators, the signal-to-noise ratio is lower since the MAC operations are computed in the analog domain. Thus, the inherent variations in IMC devices and peripheral circuits result in noisy digital output MAC values, as shown in figure 1. Figure 1(a) shows the variations in the ADC outputs for different MAC values in [7] and figure 1(b) shows the deviation of observed ADC outputs from the expected ADC outputs in [8]. These erroneous ADC outputs induce a degradation in the accuracy of the DNNs whose computations are performed in IMC hardware [7, 13–17]. For example, inference accuracy degradation of up to ∼7% for the CIFAR-10 dataset was reported when baseline DNNs are evaluated on the noisy IMC ASICs of [13], where all 128 rows of the IMC crossbar array are activated simultaneously.

Some recent works have attempted to restore the IMC inference accuracy loss with the help of software-based training methods where noise is injected at the weight level [18–20]. The main drawback of these works is that they do not use actual IMC hardware noise, but only use approximate noise models that follow the Gaussian distribution. This leads to a mismatch in the training and inference noise, and hence DNNs trained with such noise models will still exhibit some degradation in IMC inference accuracy. This is explained in detail in the next section. Furthermore, as these works inject noise into individual weights only, the training algorithm is not aware of the ADC quantization noise present in actual IMC hardware. In [21], the authors attempted to closely match the training and inference noise by using the IMC computation flow in the forward pass of the DNN, but their hardware aware training scheme is limited to approximate noise models for the ADC quantization.

On the other hand, adversarial attacks is one of the key concerns for the robustness of DNNs, since manipulating the inputs/weights of DNNs by tiny amounts has effectively reduced the inference accuracy to ∼0%. For example, the projected gradient descent (PGD) [22] and fast gradient signed method [23] algorithms use back-propagated gradients to determine the optimal adversarial input pixel perturbations that are indistinguishable by a naked eye. In earlier works, the obfuscation of gradients by means of quantizing the DNN weights and activations was floated as a viable defense against adversarial input attacks. However, the backward-pass differential approximation technique exposes this false sense of security [24] after bypassing the issue of obfuscated gradients and reducing the accuracy of low-precision quantized DNNs to ∼0% using adversarial input attacks.

In this work, we propose an IMC noise-aware training method for obtaining DNNs with high inference accuracy and enhanced robustness against adversarial attacks. We use actual noise data that is measured from the resistive RAM (RRAM) IMC prototype chips reported in [13]. Our proposed IMC noise-aware training method results in better IMC hardware inference accuracy when compared to evaluating DNNs that are trained with software-level Gaussian noise injection methods. In addition, the same proposed training method helps improve adversarial robustness by minimizing the accuracy loss caused by the input pixel perturbations using the noisy partial sum quantization which occurs during IMC hardware inference. We present results obtained on ResNet-18, AlexNet, VGG, and MobileNet DNNs for CIFAR-10 dataset, and also for three different precisions of 1-bit, 2-bit, and 4-bit for ResNet-18.

2. Background and related works

2.1. RRAM based in-memory computing

Recently, NVM based IMC prototype chips have been reported in [13, 16, 25, 26] and SRAM-based IMC prototype chips have been reported in [10–12]. Although the NVM technology

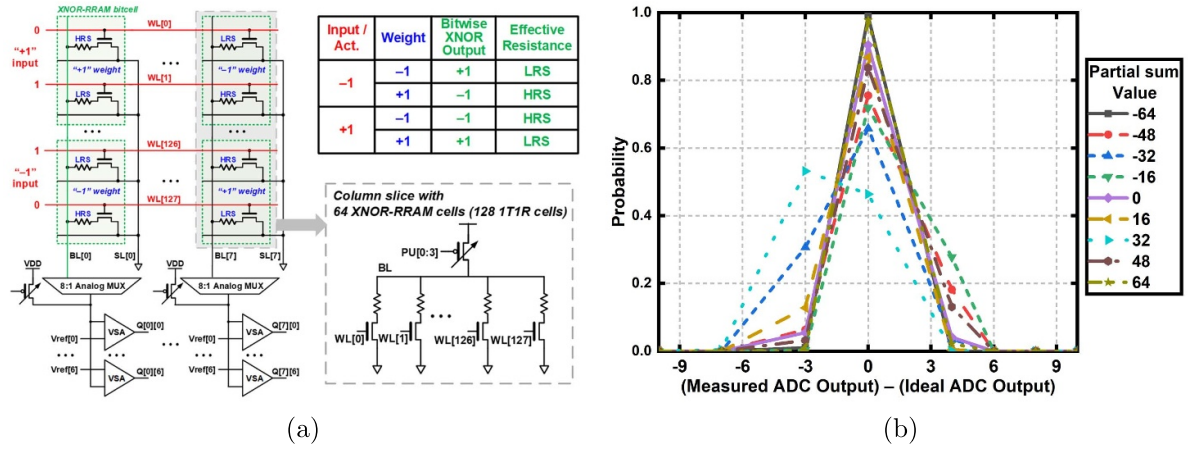

Figure 2. (a) Operation of the XNOR-RRAM IMC hardware, and (b) illustration of the actual RRAM hardware noise. (a) © [2019] IEEE. Reprinted, with permission, from [13].

is less mature than CMOS, they have advantages such as CMOS-compatibility, low power consumption, non-volatility, and most importantly, exhibit higher density compared to SRAMs. To that end, we focus on RRAM-based IMC design in this work.

Figure 2(a) shows the key operations of the XNOR-RRAM chip reported in [13]. Each 1T1R cell can be programmed to either a low resistance state (LRS) or a high resistance state (HRS). According to the combination of each 1T1R cell resistance, two 1T1R cells of the successive rows can effectively store a $+1$ or -1 weight value. The XNOR operation between the input/activation and the weight follows the function described in the table in figure 2(a). Thus, by suitably encoding the HRS and LRS of the RRAM devices, and providing the suitable values on the word lines, we can effectively perform the matrix vector multiplication of binary weights and activations.

2.2. Hardware-aware DNN training for accurate DNN inference with IMC hardware

Although the IMC technology achieves high energy efficiency, when DNNs trained in software are deployed on IMC hardware, accuracy degradation can occur due to limited ADC precision, variations in the IMC devices, ambient conditions, and transistor non-linearity [11, 13, 19, 25, 27, 28]. Several recent works have attempted to address these particular issues, and representative NVM-based works are described below.

In [29], the authors present DNNs that are tolerant to ambient temperature fluctuations. However, additional area and energy overhead are incurred in the IMC hardware due to the addition of thermal reference cells. A quantization-aware DNN training scheme was proposed in [28] which considered input and weight quantization, RRAM-based convolution, and ADC quantization. However, only up to 36 rows are activated simultaneously for IMC to limit the accuracy degradation, and still, $>2\%$ accuracy loss is reported for the CIFAR-10 dataset. In [19], the authors measured the variation in the conductance of their 11-state phase-change memory devices, and the measurements were used indirectly to inject noise via a Gaussian distribution during DNN training. Furthermore, since the conductance data of individual devices are used to inject Gaussian noise for individual weights of DNNs, this method does not consider the IMC crossbar structure or ADC quantization.

2.3. Mismatch between Gaussian and IMC hardware noise

In [19], the authors experiment injecting noise in four different ways: additively to the weights, multiplicatively to the weights, additively to the input activations of each layer, and additively to pre-activations (full sum) of each layer. Furthermore, only Gaussian distributions are used to sample the noise values in these experiments. The authors of [19] conclude that injecting Gaussian noise additively to individual weights produced the best accuracy results. In [19], partial sum level (corresponding to the computation on actual IMC crossbar) Gaussian noise injection has not been explored.

In addition, when DNNs are evaluated on an IMC crossbar, the layer computation is a composite of several partial sum evaluations, each of which are quantized by a non-ideal ADC present at the periphery of the IMC hardware. Thus, the above techniques do not ideally emulate the lumped noise observed at the output of the ADC which contains the device noise, bitline noise, and the ADC quantization noise.

Furthermore, in section 4, we show that DNN training based on injecting noise at the partial sum level using Gaussian distributions results in sub-optimal DNN inference accuracy for actual IMC chips. If we treat the IMC hardware as a black-box, the difference between an observed and expected ADC output represents aggregate noise from all the noise sources present in the IMC hardware, and it is not necessarily Gaussian, as shown in figure 2(b). This results in a noise mismatch during Gaussian noise-based training and actual IMC inference. The mismatch is further aggravated by the inter/intra-chip variations of this noise. Therefore, if the DNNs are trained with software-based noise models, usually with Gaussian distributions, it does not result in optimal

IMC inference accuracy results. An analysis of the mismatch between these different types of noise injected at the partial sum level is presented below.

If P is a partial sum between the inputs and weights in one column of the IMC crossbar, in the ideal case, we get a quantized partial sum QP at the ADC output as:

$$QP = IF(P); Loss_{ideal} = f(W, IF), \quad (1)$$

where $IF()$ is the ideal ADC transfer function, and the loss corresponding to quantization noise is a function of IF and weights, W. If we use Gaussian noise to distort the quantized partial sum during training, the equation would be:

$$GQP = QP + N(\mu, \sigma); Loss_{noise} = f(W_1, IF, N(\mu, \sigma)), \quad (2)$$

where $N(\mu, \sigma)$ is a Gaussian distribution with mean μ and standard deviation σ, and the loss corresponding to the noisy quantization is a function of N and weights, W_1. On the other hand, if we use the noisy partial sum quantization scheme obtained from IMC chip measurement data, we get:

$$NQP = NF(P); Loss_{noise} = f(W_2, NF), \quad (3)$$

where $NF()$ is the measured noisy ADC transfer function of the IMC hardware, and the loss corresponding to the noisy quantization is a function of NF and weights, W_2. After training, $W1$ *and* $W2$ will converge to different sets of values. During inference, the accuracy will be a function of weights, W, and real-world hardware noise, NF. Hence, if we use real-world hardware noise during training and inference (NF), we get optimal results compared to other approximate model-based noise-aware training.

2.4. Adversarial input attacks

DNNs have been shown to be vulnerable to small input pixel perturbation, popularly known as *adversarial examples attack* [22, 23, 30]. These attacks can be broadly classified as white-box [22, 23] and black-box adversarial attacks [30]. The PGD algorithm is a powerful white-box adversarial attack proposed in [22], which iteratively perturbs the input image pixels to maximize the corresponding loss to each perturbation (L_∞-based), by exploiting the signs of the gradients. The process follows the below equation:

$$\hat{x}^{t+1} = \hat{x}^t + \alpha \cdot sign(\nabla_x \hat{\mathcal{L}}(f(\hat{x}^t; \theta), y)), \quad (4)$$

where $f(;)$ is the DNN inference function with parameters θ, α is the step size, and $\hat{x} \in [0,1]$ for normalized input. PGD attack [22] generates universal and strong adversary among the first order approach (i.e. attacks rely on only first order gradient information) by adding the gradient sign of the loss function $\mathcal{L}$ with regard to the input x.

The white-box adversarial input attacks require smooth back-propagation of gradients from the output side to the input side of the DNN, without any obfuscation of the gradients. However, the DNN activation/weight quantization often use non-linear functions in the forward path of the DNNs, which gives rise to *gradient obfuscation* [24]. The ADC quantization of MAC values in IMC hardware is another example of such non-linear functions. This non-linearity correlates with non-differentiability, and hinders the true approximation of the gradients. This leads to poor white-box attack performance, and a false sense of adversarial robustness. The black-box adversarial attacks can be employed to bypass this gradient obfuscation and evaluate adversarial robustness in the presence of gradient obfuscation. To perform this attack, a full precision substitute model is trained to mimic the functionality of the DNN under attack, where no gradient obfuscation is present. The substitute model is attacked to obtain strong adversarial examples, which are then used to evaluate the inference accuracy of the target models. This black-box attack is illustrated in figure 3.

The robustness of DNNs to adversarial input attacks is improved by training them with both clean and adversarial input images [22]. This optimizes both the adversarial and clean image losses simultaneously, and the adversarial optimization follows the equation: $Min_\theta\{Max_{x'} \mathcal{L}(f(\hat{x}; \theta), y)\}$. Here, the inner maximization generates adversarial samples $\hat{x}$ by maximizing the loss with regard to label y, and the outer minimization trains the DNN parameters θ using the adversarial samples forming a min-max optimization problem.

Recent works have improved the adversarial robustness of DNNs by injecting noise into the DNN weights during training, as noise injection acts as a regularizer, and prevents DNNs from over-fitting [31–33]. However, noise injection combined with adversarial training also causes gradient obfuscation. Alternatively, some works have quantized the DNN parameters during training to use this inherent introduction of gradient obfuscation as an adversarial defense mechanism [34, 35]. Although quantization of DNN parameters improves adversarial robustness, it alone is not sufficient to defend against adversarial attacks. Hence, we studied the effect of actual IMC hardware noise-aware training along with adversarial training on the adversarial robustness of DNNs by subjecting them to black-box PGD adversarial attacks.

3. Proposed IMC hardware noise-aware DNN training and inference

In IMC hardware, the MAC values (partial sums) are converted into an M-bit binary number, where the range of quantization is 2^M discrete levels. The value of M is desired to be as low as possible in order to minimize the ADC overhead. Due to the IMC variations, the MAC values from the DNN computation that have the same value could result in different ADC outputs. The limited ADC precision, coupled with other IMC variations results in noisy computations, and making the training algorithm aware of such a noise will improve DNN robustness to IMC hardware noise. In this work, we used the noise data that is measured at a supply voltage of 1.2 V from the XNOR-RRAM IMC prototype chips reported in [13].

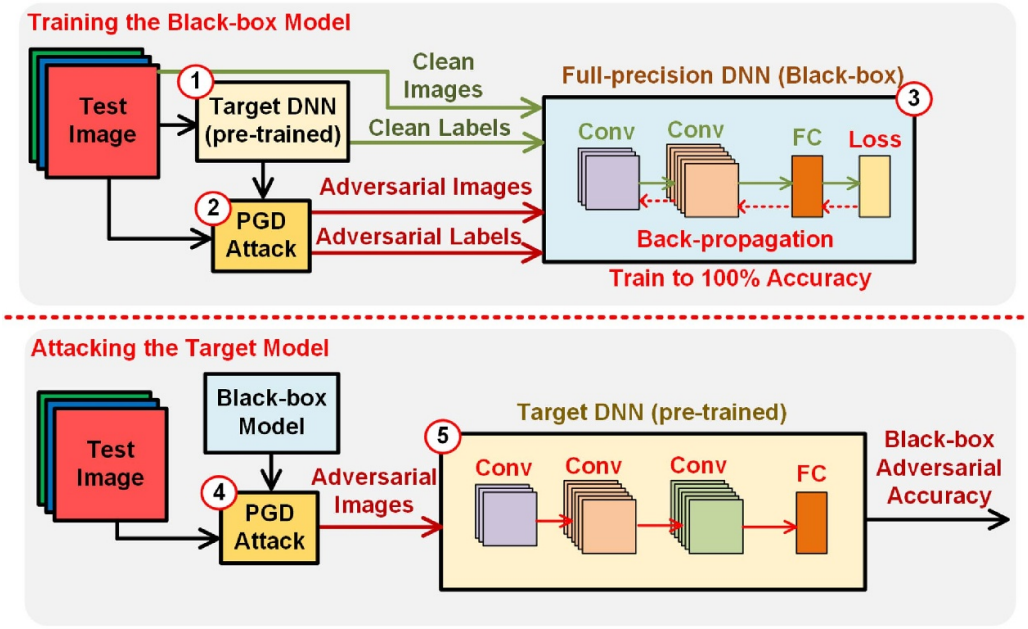

Figure 3. Illustration of the black-box PGD attack method.

3.1. IMC hardware and quantization noise

In the XNOR-RRAM macro [13] two bit-cells from successive rows are used to store one weight value, and all the 128 rows are activated simultaneously to perform 64-input MAC operations between signed binary activations and weights (−1 and +1). The MAC value is expected to be between −64 to +64, and is measured as an analog voltage, V_{BL}. This analog voltage is digitized by a 3-bit flash ADC to one of the 8 possible binary output levels, i.e. [0, 1,…, 7]. These levels can be further encoded as different actual quantization values, namely, [−17, −11, −7, −3, 0, 4, 8, 14]. This encoding scheme was finalized by doing several software simulations and analyses of the partial sum distributions in various DNNs during training. We found that less than 0.1% of the possible partial sum values are either less than −32 or greater than +32. Hence, we clipped the possible partial sum values to −32 or +32 before performing the ADC quantization.

The IMC noise can also be visualized in another way. We can define the ADC quantization error as the difference between the measured ADC output and the ideal ADC output. Inferred from the corresponding conditional probability table, figure 2(b) shows a distribution of the ADC quantization error in the XNOR-RRAM chips measured at 1.2 V for, where each curve represents the error distribution for a particular partial sum value in the range of −64 to +64. These errors follow a Gaussian distribution for a majority of the MAC values, and hence an approximate Gaussian curve-fit was obtained with a mean of 0.146 and a standard deviation of 2.36. The fitted Gaussian model depicts a MAC-value-independent ADC quantization error distribution, which can approximate the XNOR-RRAM hardware noise.

3.2. DNN inference with IMC hardware emulation

Figure 4 shows an illustration of the computation flow that is used to evaluate a fully connected DNN layer with 128 input neurons on XNOR-RRAM IMC hardware with 128 bit-cells per column. We can perform one 64-input MAC operation per column at a time because two RRAM devices in the same column are used to store one weight value. Therefore, the computation is broken down into two groups of 64 inputs. The quantized partial sums are then accumulated to obtain the quantized full-sum. Additionally, for multi-bit DNN evaluation, multi-bit weights are split across multiple columns of the IMC array and multi-bit activations are fed to the IMC array over multiple cycles to perform bit-serial processing. The partial sums are then accumulated with proper binary-weighted coefficients depending on the bit positions of the sub-activations/weights, and the full sum for a given neuron in the DNN layer is obtained.

Therefore, for every 64-input MAC operation, we use IMC chip measurement results with a sampling method described below to quantize the partial sum. Accumulation of such 64-input partial sums and other non-MAC operations are performed via digital simulation. First, to characterize the noisy quantization behavior for the XNOR-RRAM chips, we performed a total of 128 000 measurements for 64-input MACs with randomized input vectors, and obtained 2D histograms between each MAC value and ADC output. A conditional probability table is obtained from this data, which reports the probability of each MAC value being quantized to different ADC outputs. Each probability table is unique to the chip being measured, the supply voltage, and ambient conditions. In this work, we used the probability data shown in figure 1(a), where it can be seen that the same bit count value can be

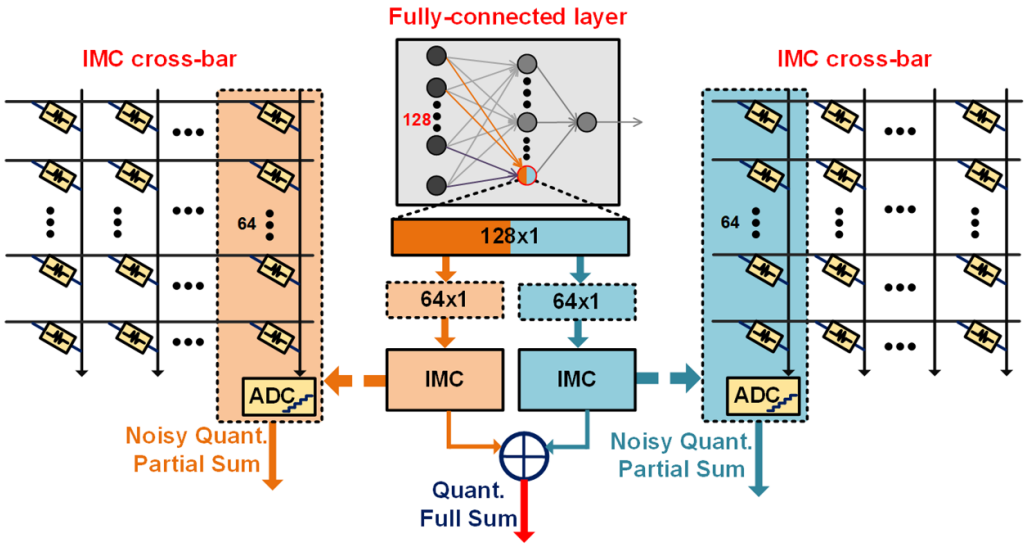

Figure 4. A fully-connected neuron with 128 inputs can be evaluated on IMC hardware with 64 rows by breaking it down into two groups.

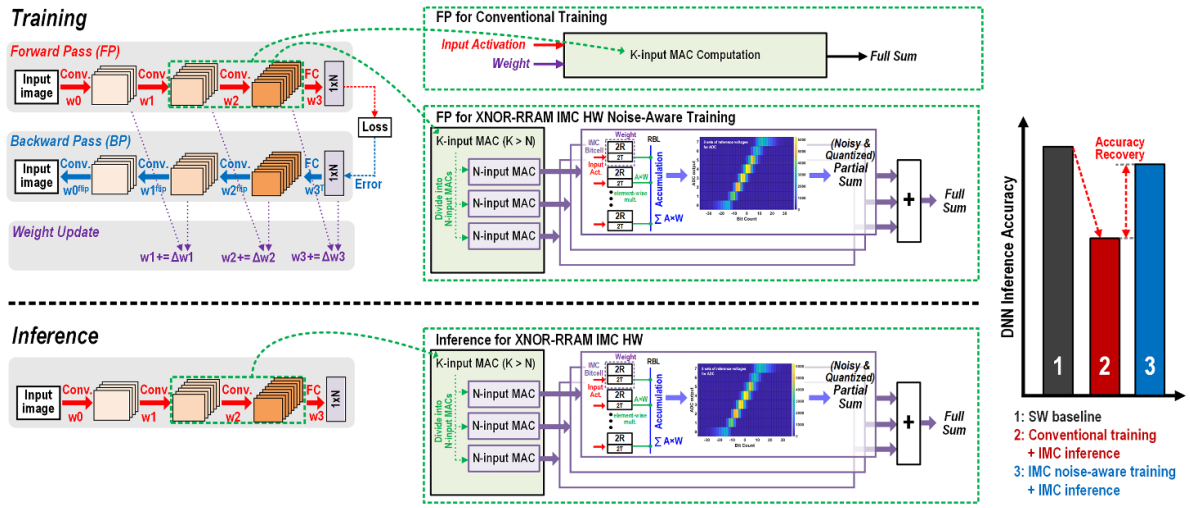

Figure 5. Illustration of the proposed IMC hardware noise-aware training and IMC inference evaluation.

transformed to more than one ADC value. Thus, to evaluate the accuracy of large DNNs, for each 64-input MAC value, we randomly 'sample' the ADC output based on the distribution in the probability table. We employ this sampling method from the chip measurement results since the small size of IMC chips prevents us from directly mapping large DNNs, and time-multiplexing the small IMC chip will make the testing time excessively long.

3.3. IMC hardware noise-aware training

In the conventional IMC works, the training algorithm was not made aware of the hardware variability and quantization noise, and the IMC inference accuracy of software-trained DNNs is affected by the above-discussed hardware noise. Hence, we performed noise-aware DNN training by injecting the measured IMC hardware noise into the forward pass during the training phase, as shown in figure 5. This enables the optimization of DNN parameters by considering the noisy partial sum quantization, resulting in better IMC hardware inference accuracy performance.

We injected the hardware noise by emulating the IMC macro's computation flow and then used the conditional probability tables to transform the smaller chunks of MAC values (i.e. partial sums) in a similar way to the actual IMC hardware, as shown in Algorithm 1. We also made this trainable by using a windowed straight-through-estimator for the backward pass.

Algorithm 1. Noise-aware DNN training.

Input: n binary inputs x_i and weights w_i
Input: IMC row-size r, cumulative noise probability matrix pt
Output: Noisy quantized MAC $Q(\sum_1^n x_i \times w_i)$
Initialize: number of chunks $c = ceil(n/r)$
Initialize: Divide the inputs and weights into c chunks, MAC value, $d = 0$.
cdf.find(cdf, x): identifies the index of the first element in cdf that is less than x
random.uniform(): returns a random float in $[0, 1]$
for $i = 1$ **to** c **do**
  partial sum $ps = \sum_1^r x_i \times w_i$
  level-probs = pt[ps]
  index = cdf.find(level-probs, random.uniform())
  qlevel = levels[index]
  Q(cdp) = qlevel
  d = d + Q(cdp)
end for
return d

4. Experiment results

We performed XNOR-RRAM IMC hardware noise-aware DNN training and evaluated ResNet-18, AlexNet, VGG, and MobileNet DNNs for the CIFAR-10 dataset. For VGG DNN, We used the model presented in [2], i.e. *input*-128*C*3-128*C*3-*P*2-256*C*3-256*C*3-*P*2- 512*C*3-512*C*3-*P*2-1024 *FC*-1024 *FC*-10 *FC*. Here, 128C3 is convolution layer with 128 channels and 3×3 kernels, P2 is 2×2 max-pooling, and 1024FC is fully-connected (FC) layer with 1024 neurons. We also evaluated ResNet-18 and VGG DNNs with three different activation/weight precisions of 1-bit, 2-bit, and 4-bit. We used measurement results from the XNOR-RRAM IMC prototype chip reported in [13]. We employed quantization-aware training [2] for low-precision DNN inference. For the proposed hardware noise-aware training, all DNNs were trained by using a batch size of 50 and the default hyper-parameters in [2]. Furthermore, the reported DNN inference accuracies are the average values obtained from five inference evaluations of the same DNN under the same noise conditions used during the proposed training process.

4.1. Improvement in IMC inference accuracy

We trained baseline DNNs where no noise was injected during training, and also trained the same DNNs by injecting the IMC hardware noise. Specifically, the following set of inference accuracies are obtained: (1) *Baseline Accuracy* represents software baseline accuracy without any noise injection, (2) *Conventional IMC Inference Accuracy* represents the DNN inference accuracy with IMC chip measurement-based evaluation on the baseline DNNs without noise-aware training, and (3) *Noise-Aware IMC Inference Accuracy* represents IMC chip measurement-based evaluation on the new DNNs trained with the proposed hardware noise injection.

We performed DNN training/inference on ResNet-18, VGG, AlexNet, and MobileNet DNNs for CIFAR-10, using the XNOR-RRAM IMC noise measured at 1.2 V supply. Initially, the weights and activations were binarized in all networks during training, except MobileNet where only the convolution layers were binarized. Later on, we evaluated multi-bit ResNet-18 and VGG DNNs using our proposed noise-aware training scheme. In all the binarized models, the inputs to the first layer were normalized to be between ±2.5 to obtain optimal training results, instead of the typically used normalization range of $[0, 1]$.

Furthermore, in order to compare the effectiveness of the proposed noise-aware training with Gaussian noise-based training, we first replaced the bit-wise probability table-based noise injection with the ideal partial sum quantization function of the IMC hardware. We then added noise drawn from a single noise model obtained from the quantization error distribution, an example of which is shown in figure 2(b). The Gaussian curve that best fits the error distribution was chosen as the single noise model, which has a mean of 0.146 and a standard deviation of 2.36.

4.1.1. Different DNN models. Figure 6(a) shows the results on the binarized DNN models reported in [2], where the proposed IMC noise-aware training helps restore the IMC hardware accuracy closer to the software baseline in all models. In all the cases, although adding Gaussian noise to the partial sums results in improvements compared to conventional IMC inference, the best results are obtained when injecting actual IMC noise during training. For ResNet-18 and VGG, the IMC hardware accuracy can be restored to <3% of the software baseline, compared to ∼3%–11% accuracy degradation of the conventional scheme. The accuracy improvement is sub-optimal in the Mobilenet case as this DNN is highly sensitive to noise because of the depth-wise and point-wise convolution layers. It is a highly optimized DNN, making it more sensitive to noise-induced variations in the computations. The scope for decreasing the influence of computation errors in the depth-wise and point-wise convolution layers is less when compared to using conventional convolution layers.

4.1.2. Different DNN precision. Figures 6(b) and (c) show the IMC hardware accuracy improvements in ResNet-18 and VGG DNNs respectively, for activation/weight precision values of 1-bit, 2-bit, and 4-bit. As we increase the DNN precision, the IMC accuracy without noise-aware training worsens because IMC hardware performs bit-wise computations in each column, and as multiple columns' ADC outputs get shifted/added to perform the multi-bit MAC operations, more noise accumulates to the final MAC value. Another cause for more accuracy degradation as we increase the DNN precision is the 64-input size limitation for the binary MACs, as a result of which more noise is added compared to a larger MAC size of say 256 inputs used in [10]. However, the proposed noise-aware training scheme restores the accuracies for 1-bit/2-bit/4-bit ResNet-18 and VGG DNNs significantly compared to the conventional training scheme.

4.1.3. Noise from different chips. We performed the same noise-aware DNN training for ResNet-18 by using three

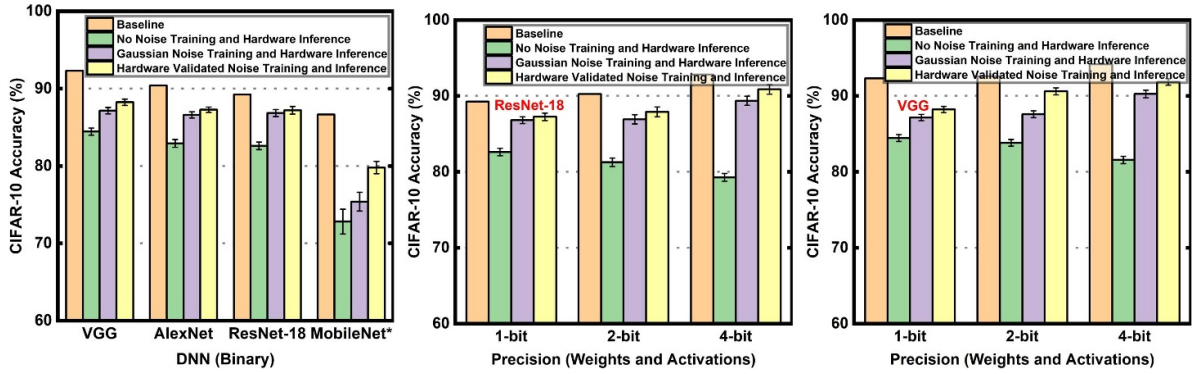

Figure 6. IMC inference accuracy along with corresponding standard deviation error bars after hardware noise-aware training across different DNN topologies shows consistent improvement.

Table 1. IMC inference accuracies for binary ResNet-18 on CIFAR-10 dataset after different noise-aware training schemes.

Baseline Binary ResNet-18 CIFAR-10 Accuracy: 89.24 ± 1.05%			
Training Inference	Conventional IMC Baseline Individual chip	Noise-aware IMC Individual chip Individual chip	Noise-aware IMC Ensemble (3 Chips) Ensemble (3 Chips)
Chip 1	82.60% ± 0.54%	87.24% ± 0.58%	
Chip 2	82.17% ± 0.62%	87.33% ± 0.6%	87.57% ± 0.6%
Chip 3	81.53% ± 0.48%	87.11% ± 0.55%	
Average	82.1% ± 0.55%	87.22% ± 0.58%	87.57% ± 0.6%
2-bit ResNet-18 CIFAR-10 Accuracy: 90.24 ± 0.53%			
Training Inference	Conventional IMC Baseline Individual chip	Noise-aware IMC Individual chip Individual chip	Noise-aware IMC Ensemble (3 chips) Ensemble (3 chips)
Chip 1	81.26% ± 0.51%	87.59% ± 0.61%	
Chip 2	80.80% ± 0.62%	87.41% ± 0.54%	87.82% ± 0.66%
Chip 3	81.37% ± 0.55%	87.64% ± 0.58%	
Average	81.14% ± 0.56%	87.55% ± 0.58%	87.82% ± 0.66%
4-bit ResNet-18 CIFAR-10 Accuracy: 92.81 ± 0.32%			
Training Inference	Conventional IMC Baseline Individual chip	Noise-aware IMC Individual chip Individual chip	Noise-aware IMC Ensemble (3 chips) Ensemble (3 chips)
Chip 1	79.26% ± 0.49%	90.86% ± 0.63%	
Chip 2	79.11% ± 0.52%	90.72% ± 0.66%	91.22% ± 0.58%
Chip 3	79.81% ± 0.51%	91.05% ± 0.68%	
Average	79.39% ± 0.51%	90.88% ± 0.66%	91.22% ± 0.58%

different noise probability tables, obtained from three different XNOR-RRAM chips. Table 1 shows that, on average, the IMC inference accuracy was improved by 5.1%, 6.41%, and 11.49% for 1-bit, 2-bit, and 4-bit ResNet-18 DNNs respective. Hence, DNNs trained with our proposed training algorithm are robust to inter-chip variations in the IMC noise as well.

4.2. Improvement in adversarial input attack robustness

We evaluated the adversarial robustness of multi-bit ResNet-18 DNNs using black-box adversarial attack and CIFAR-10 dataset, as shown in figure 7. As shown in figure 7(a), the PGD attack accuracy is improved by up to ∼6% in comparison to the conventionally trained baseline by adding XNOR-RRAM IMC noise from [13] to the DNN training and inference process. The PGD attack accuracy is further improved by using adversarial training, and significant accuracy improvement is observed across all the DNNs in figure 7(b) compared to those in figure 7(a). Hence, the IMC hardware noise and partial sum quantization act as an inherent regularizer. This improves the tolerance of the DNN to the deviation in the partial sum values, which are in-turn caused by perturbations in the input pixels.

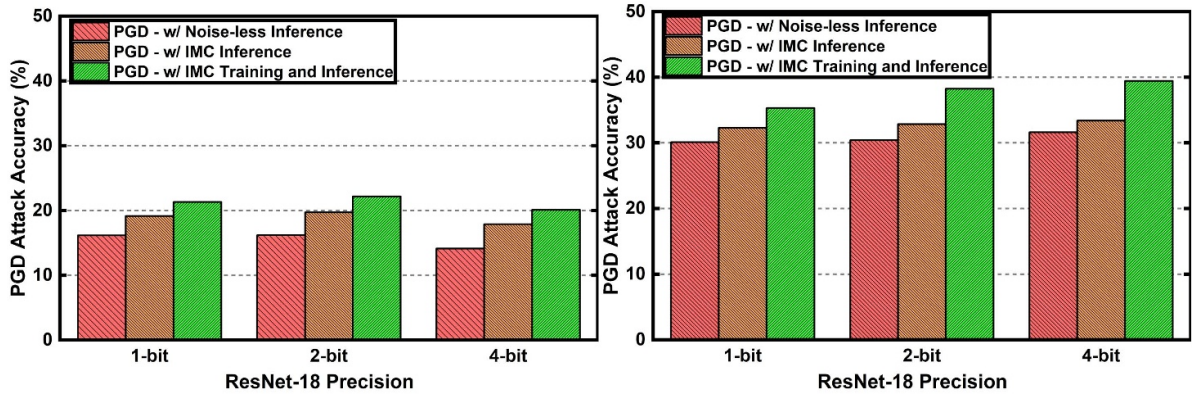

Figure 7. Black-box adversarial accuracy with IMC-noise-aware training based on IMC measurements at 1.2 V [13]. The average results across 5 runs of PGD attacks are shown for (a) DNNs trained with only clean images and (b) DNNs trained with clean and adversarial images (i.e. adversarial training).

Table 2. Comparison of our work with two other related works, where we demonstrate better capability to restore DNN IMC inference accuracy after noise-aware training.

Training noise	XNOR-RRAM IMC inference accuracy (%)					
DNN model	ResNet-18			VGG		
Precision	1-bit	2-bit	4-bit	1-bit	2-bit	4-bit
Software baseline	89.24	90.24	92.81	92.3	92.68	93.5
No noise	82.6	81.26	79.26	84.45	83.33	81.39
Joshi *et al* [19]	85.26	85.48	86.92	86.33	86.63	88.19
Zhou *et al* [36]	86.16	86.2	87.14	86.4	86.58	88.83
XNOR-RRAM (our work)	87.24	87.89	90.9	88.2	90.6	91.85

Therefore, we can also improve the adversarial robustness of DNNs using IMC hardware noise-aware training.

4.3. Comparison to relevant works

We compare our work with two relevant works [19, 36]. In both of these works, noise is added to individual weights during DNN training, and it differs from actual IMC hardware noise, as discussed in section 2.3. Thus, although these works tend to improve the IMC inference accuracy compared to the software baselines trained without any noise injection, some IMC inference accuracy degradation is expected. Our proposed noise-aware training scheme matches well with actual IMC hardware noise, and therefore more effectively improves IMC inference accuracy.

We performed noise-aware training using the $\eta_{tr} = \eta_{inf}$ combination with a value of 0.138 for the work of [19], and a value of 0.069 for the work of [36]. These values were chosen so that the noise remains the same during training and inference, and also so that the noise intensity remains similar to that of the XNOR-RRAM noise. The maximum and minimum ADC output values on which noise is applied (corresponding to the XNOR-RRAM design [13]) are $+17$ and -17, respectively. The standard deviation of the XNOR-RRAM chip noise is 2.36. We obtain the aforementioned η values by substituting these values into the noise formulae of $\frac{\sigma_{noise}}{W_{max}} = \eta$ provided by [19] and $\sigma^{l}_{noise} = \eta \times (W^{l}_{max} - W^{l}_{min})$ provided by [36].

Table 2 shows that our proposed IMC noise-aware training algorithm achieves better IMC inference accuracy compared to the two related works across ResNet-18 and VGG DNNs with 1-bit, 2-bit, and 4-bit precision. Furthermore, the amount of net noise for IMC inference with higher precision DNNs increases since error accumulates during bit-by-bit operations, and this results in larger accuracy degradation. Hence, by using IMC hardware noise-aware training, the accuracy loss can be recovered more effectively compared to the two relevant works in 2-bit and 4-bit DNNs.

5. Conclusion

In this work, we presented a new hardware noise-aware DNN training scheme to improve the DNN inference accuracy and adversarial robustness of RRAM-based IMC hardware. By injecting noise measured from RRAM IMC prototype chips during DNN training, the RRAM IMC inference accuracy for CIFAR-10 dataset improved by up to 8%. We have also shown that the DNNs can be made more robust to adversarial input attacks by using our proposed noise-aware training scheme, and robustness improvements of up to 6% can be obtained

by just introducing IMC hardware noise during DNN training and inference. If adversarial training is used alongside IMC noise-aware training, the robustness can be improved by up to 25%. The techniques proposed in this work can be utilized as a general DNN framework to integrate empirically measured hardware errors that are more complicated than simple Gaussian noise and exhibit data dependence, towards achieving high accuracy and robustness.

Data availability statement

The data generated and/or analyzed during the current study are not publicly available for legal/ethical reasons but are available from the corresponding author on reasonable request.

Acknowledgments

This work was supported in part by NSF Grant Nos. 1652866, 1919147, and C-BRIC and ASCENT, two of the six centers in JUMP, a SRC program sponsored by DARPA.

ORCID iDs

Sai Kiran Cherupally https://orcid.org/0000-0002-6305-1935
Injune Yeo https://orcid.org/0000-0002-4596-6170
Shimeng Yu https://orcid.org/0000-0002-0068-3652
Jae-Sun Seo https://orcid.org/0000-0002-4551-7789

References

[1] Choi J *et al* 2019 Accurate and efficient 2-bit quantized neural networks *Conf. Machine Learning and Systems (MLSys)*
[2] Hubara I, Courbariaux M, Soudry D, El-Yaniv R and Bengio Y 2016 Binarized neural networks *Advances in Neural Information Processing Systems* pp 4107–15
[3] Park E and Yoo S 2020 PROFIT: a novel training method for sub-4-bit mobilenet models *ECCV*
[4] Chen Y-H, Krishna T, Emer J and Sze V 2017 Eyeriss: an energy-efficient reconfigurable accelerator for deep convolutional neural networks *IEEE J. Solid-State Circuits (JSSC)* **52** 127–38
[5] Sim J, Lee S and Kim L 2020 An energy-efficient deep convolutional neural network inference processor with enhanced output stationary dataflow in 65-nm CMOS *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **28** 87–100
[6] Zimmer B *et al* 2020 A 0.32–128 TOPS, scalable multi-chip-module-based deep neural network inference accelerator with ground-referenced signaling in 16 nm *IEEE J. Solid-State Circuits* **55** 920–32
[7] Yin S, Sun X, Yu S and Seo J-S 2020 High-throughput in-memory computing for binary deep neural networks with monolithically integrated rram and 90-nm cmos *IEEE Trans. Electron Devices* **67** 4185–92
[8] Li W, Huang S, Sun X, Jiang H and Yu S 2021 Secure-rram: a 40 nm 16 kb compute-in-memory macro with reconfigurability, sparsity control and embedded security *2021 IEEE Custom Integrated Conf. (CICC)* (IEEE) pp 1–2
[9] Si X *et al* 2020 A Twin-8T SRAM computation-in-memory unit-macro for multibit CNN-based AI edge processors *IEEE J. Solid-State Circuits* **55** 189–202
[10] Yin S, Jiang Z, Seo J and Seok M 2020 XNOR-SRAM: in-memory computing SRAM macro for binary/ternary deep neural networks *IEEE J. Solid-State Circuits* **55** 1733–43
[11] Dong Q, Sinangil M E, Erbagci B, Sun D, Khwa W, Liao H, Wang Y and Chang J 2020 A 351TOPS/W and 372.4GOPS compute-in-memory SRAM Macro in 7nm FinFET CMOS for machine-learning applications *IEEE Int. Solid-State Conf. (ISSCC)* pp 242–4
[12] Jiang Z, Yin S, Seo J and Seok M 2020 C3SRAM: an in-memory-computing SRAM macro based on robust capacitive coupling computing mechanism *IEEE J. Solid-State Circuits* **55** 1888–97
[13] Yin S *et al* 2019 Monolithically integrated RRAM-and CMOS-based in-memory computing optimizations for efficient deep learning *IEEE Micro* **39** 54–63
[14] Yu S, Shim W, Peng X and Luo Y 2021 RRAM for compute-in-memory: from inference to training *IEEE Trans. Circuits Syst.* I **68** 2753–65
[15] Liu Q *et al* 2020 33.2 a fully integrated analog RERAM based 78.4tops/w compute-in-memory chip with fully parallel mac computing *2020 IEEE Int. Solid State Conf. (ISSCC)* pp 500–2
[16] Chen Y 2020 Reram: history, status and future *IEEE Trans. Electron Devices* **67** 1420–33
[17] Zhang Y, Huang P, Gao B, Kang J and Wu H 2020 Oxide-based filamentary rram for deep learning *J. Phys. D: Appl. Phys.* **54** 083002
[18] He Z, Lin J, Ewetz R, Yuan J-S and Fan D 2019 Noise injection adaption: end-to-end RERAM crossbar non-ideal effect adaption for neural network mapping *Proc. 56th Annual Design Conf. 2019* pp 1–6
[19] Joshi V *et al* 2020 Accurate deep neural network inference using computational phase-change memory *Nat. Commun.* **11** 1–13
[20] Zhou A, Yao A, Guo Y, Xu L, and Chen Y 2017 Incremental network quantization: towards lossless CNNS with low-precision weights (arXiv:1702.03044)
[21] Gokmen T, Rasch M J and Haensch W 2019 The marriage of training and inference for scaled deep learning analog hardware *2019 IEEE Int. Electron Devices Meeting (IEDM)* (Piscataway, NJ: IEEE) pp 22–5
[22] Madry A *et al* 2018 Towards deep learning models resistant to adversarial attacks *ICLR*
[23] Goodfellow I J *et al* 2014 Explaining and harnessing adversarial examples (arXiv:1412.6572)
[24] Athalye A *et al* 2018 Obfuscated gradients give a false sense of security: circumventing defenses to adversarial examples *ICML*
[25] Xue C *et al* 2020 A 22nm 2Mb ReRAM compute-in-memory macro with 121-28TOPS/W for multibit MAC computing for tiny AI edge devices *IEEE Int. Solid State Conf. (ISSCC)* pp 244–6
[26] Song L, Qian X, Li H and Chen Y 2017 Pipelayer: a pipelined RERAM-based accelerator for deep learning *2017 IEEE Int. Symp. High Performance Computer Architecture (HPCA)* (IEEE) pp 541–52
[27] Gonugondla S K, Kang M and Shanbhag N R 2018 A variation-tolerant in-memory machine learning classifier via on-chip training *IEEE J. Solid-State Circuits* **53** 3163–73
[28] Wei W *et al* 2020 A relaxed quantization training method for hardware limitations of resistive random access memory

(ReRAM)-based computing-in-memory *IEEE J. Explor. Solid-State Comput. Devices Circuits* **6** 45–52
[29] Joardar B K, Doppa J R, Pande P P, Li H and Chakrabarty K 2020 AccuReD: high accuracy training of CNNS on ReRAM/GPU heterogeneous 3D architecture *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **40** 971–84
[30] Papernot N *et al* 2017 Practical black-box attacks against machine learning *ACM Conf. Computer and Commun. Security*
[31] He Z, Rakin A S and Fan D 2019 Parametric noise injection: trainable randomness to improve deep neural network robustness against adversarial attack *Proc. Conf. Computer Vision and Pattern Recognition* pp 588–97
[32] Srivastava N *et al* 2014 Dropout: a simple way to prevent neural networks from overfitting *J. Mach. Learn. Res.* **15** 1929–58
[33] Lecuyer M *et al* 2018 On the connection between differential privacy and adversarial robustness in machine learning (arXiv:1802.03471)
[34] Lin J *et al* 2019 Defensive quantization: when efficiency meets robustness *ICLR*
[35] He Z *et al* 2020 Defending and harnessing the bit-flip based adversarial weight attack *IEEE CVPR*
[36] Zhou C, Kadambi P, Mattina M and Whatmough P N 2020 Noisy machines: understanding noisy neural networks and enhancing robustness to analog hardware errors using distillation (arXiv:2001.04974)