Leveraging Noise and Aggressive Quantization of In-Memory Computing for Robust DNN Hardware Against Adversarial Input and Weight Attacks

Sai Kiran Cherupally[1], Adnan Siraj Rakin[1], Shihui Yin[1], Mingoo Seok[2], Deliang Fan[1], Jae-sun Seo[1]
[1]*Arizona State University*, [2]*Columbia University*

2021 58th ACM/IEEE Design Automation Conference (DAC) | 978-1-6654-3274-0/21/$31.00 ©2021 IEEE | DOI: 10.1109/DAC18074.2021.9586233

***Abstract*—In-memory computing (IMC) substantially improves the energy efficiency of deep neural network (DNNs) hardware by activating many rows together and performing analog computing. The noisy analog IMC induces some amount of accuracy drop in hardware acceleration, which is generally considered as a negative effect. However, in this work, we discover that such hardware intrinsic noise can, on the contrary, play a positive role in enhancing adversarial robustness. To achieve that, we propose a new DNN training scheme that integrates measured IMC hardware noise and aggressive partial sum quantization at the IMC crossbar. We show that this effectively improves the robustness of IMC DNN hardware against both adversarial input and weight attacks. Against black-box adversarial input attacks and bit-flip weight attacks, DNN robustness has improved by up to 10.5% (CIFAR-10 accuracy) and 33.6% (number of bit-flips), respectively, compared to conventional DNNs.**

***Index Terms*—in-memory computing, adversarial attack, adversarial robustness, noise injection, low-precision quantization**

I. Introduction

Deep neural networks (DNNs) have shown substantial success for many practical applications, e.g., image/speech recognition, autonomous driving, etc., achieving high accuracy aided by deep and complex network structures. While many works investigated DNN model size reduction [1], DNNs still require a very large number of computation and memory accesses. As a means to address such computation/memory challenges, in-memory computing has been proposed and has shown promising energy-efficiency numbers [2]–[6]. While IMC substantially improves the energy-efficiency of multiply-and-accumulate (MAC) operations in DNNs, the noise margin is lower due to the analog nature of computing and noise/variability, which led to a certain amount of accuracy degradation in the demonstrated IMC silicon works.

On the other hand, the vulnerability of DNNs against adversarial attacks has been an important issue, where adversaries can manipulate the inputs/weights of DNNs by small amounts and significantly lower the inference accuracy. Many prior works have shown that the performance of DNNs can be severely degraded by modifying the inputs of DNNs by a small amount using adversarial algorithms such as PGD [7] and FGSM [8]. These algorithms iteratively analyze the gradients at different locations in the network topology and use DNN optimization functions to identify the suitable magnitude of change in the input pixels, so that the DNN classifies the input incorrectly. A few early defense works claimed to be robust against attacks such as PGD, but it was later reported that the robustness was obtained mainly due to the presence of obfuscated gradients [9], e.g., in quantized DNNs. The issue of obfuscated gradients was circumvented to an extent using the backward-pass differentiable approximation (BPDA) technique. Hence, DNNs are vulnerable to adversarial input attacks, even if they are quantized to low-precision.

In addition, adversarial weight attacks have been reported [10]–[12], where the attacker iteratively identifies the most vulnerable bits of the weights in all DNN layers that lead to large accuracy loss. In [10], the accuracy of 8-bit DNNs was reduced to below a random guess by only flipping tens of bits in the entire model. These attacks make the DNN hardware that stores DNN weights and biases vulnerable.

Several recent works have proposed noise-injection techniques [13]–[16] to defend against adversarial attacks. Parametric noise injection [13] involves trainable Gaussian noise into the activations or weights of each DNN layer to improve the adversarial robustness. However, most prior works have employed synthetic or Gaussian noise to perturb the activations/weights of DNNs to improve robustness. Although works such as [17] have used specially designed software modules that emulate different degrees of IMC hardware noise, involving actual IMC hardware noise has not been performed. Furthermore, the effect of partial sum quantization at the IMC crossbar has not been investigated for adversarial robustness.

In this work, we investigate employing the actual measured hardware noise from IMC prototype chips [3], [5] towards enhancing the robustness of DNNs against both adversarial input attacks and weight attacks. Using the input-splitting technique [18], [19], we also evaluate the effect of aggressively quantizing the partial sums obtained from IMC crossbars on the adversarial robustness. For adversarial input attacks, we performed adversarial training [7] with a continually differentiable CELU activation function [16], [20] for DNNs with 1-bit, 2-bit, and 4-bit activation/weight precision values. We mapped all MAC operations in convolution and fully-connected layers of such pre-trained DNN models with IMC hardware designs for inference, and also investigated injecting IMC hardware noise during the DNN training process and evaluated the adversarial robustness. For adversarial weight attacks, we evaluate the effect of IMC hardware noise and

aggressive partial sum quantization via input-splitting towards the robustness against bit-flip attacks (BFA) [10], [11].

We achieve up to 10% improvement in the classification accuracy under black-box adversarial attack when IMC hardware noise and adversarial examples were used to train and test DNNs against adversarial inputs. We also show that introducing IMC noise into a conventionally trained DNN during inference leads to no degradation or even 2% improvement in adversarial accuracy. Furthermore, the input-split DNNs with aggressive partial sum quantization improved the robustness against BFA by up to 30% compared to the conventionally trained DNNs. The main contributions of this work are:

- Black-box adversarial attacks with IMC noise injection during training and testing of DNNs.
- Robustness improvement by injecting noise from actual IMC prototype chips during DNN training.
- Robustness improvement by using CELU activation function and IMC noise for DNN inference.
- Robustness improvement by using input-splitting and aggressive partial sum quantization.

II. Background and Related Works

A. SRAM-based In-Memory Computing Hardware Designs

In IMC systems, DNN weights are stored in a crossbar structure, and analog computation is performed typically by applying activations as the voltage from the row side and accumulating the bitwise multiplication result via analog voltage/current on the column side. The analog voltage/current values are quantized into digital values by analog-to-digital converters (ADCs) at the crossbar periphery. This way, vector-matrix multiplication of activation vectors and the stored weight matrices can be computed in a highly parallel manner without reading out the weights.

Both SRAM based IMC [2]–[6] and non-volatile memory (NVM) based IMC [21] have been presented in the literature. While NVM devices have density advantages compared to SRAMs, the availability of embedded NVMs in scaled CMOS technologies is limited, and peripheral circuits such as ADCs often dominate the area. Also, several device non-idealities such as low on/off ratio, endurance, relaxation, etc., pose challenges for robust NVM IMC and large-scale integration. On the other hand, SRAM has a very high on/off ratio and the SRAM IMC scheme can be implemented in any latest CMOS technology. To that end, we focus on SRAM IMC designs in this paper. SRAM IMC schemes can be broadly categorized into resistive and capacitive IMC. Resistive IMC uses the resistive pull-down/pull-up of transistors in the SRAM bit-cell [2]–[4], while capacitive IMC employs additional capacitors in the bit-cell to compute MAC operations via capacitive coupling [5] or charge sharing [6].

Fig. 1 shows the design and operation of representative resistive SRAM IMC called "XNOR-SRAM" [3] and capacitive SRAM IMC termed "C3SRAM" [5] designs. In XNOR-SRAM, the binary multiplication (XNOR) between activations driving the rows and weights stored in 6T SRAM is implemented by the complimentary pull-up/pull-down circuits of

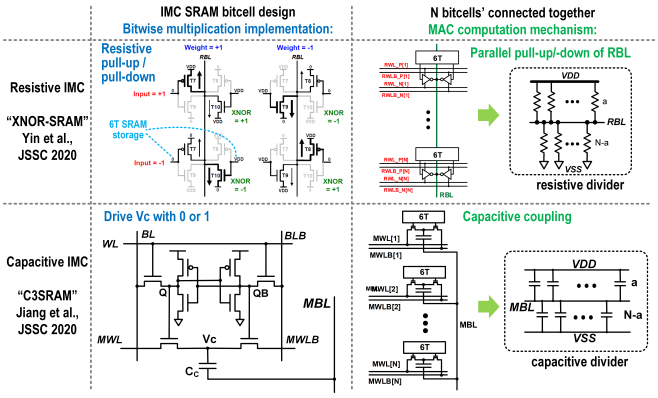


Fig. 1. Design and operation of SRAM-based resistive IMC and capacitive IMC. Adapted from [3], [5].

four additional transistors. In C3SRAM, an additional metal-oxide-metal (MOM) capacitor is introduced per bit-cell to perform MAC operations via capacitive coupling. For resistive and capacitive IMC designs, each bit-cell's bitwise multiplication result is accumulated onto the analog bit-line voltage by forming a resistive and a capacitive divider, respectively.

B. Adversarial Input and Weight Attacks

The security analysis of DNNs is dominated by the adversarial input noise attack popularly known as *adversarial examples attack* [7]–[9]. Adversarial input attacks can be classified into two major categories: white-box and black-box attacks. In a white-box attack (e.g., PGD [7], FGSM [8]), the adversary has complete knowledge about DNN inputs, architectures and gradients. In contrast, the black-box attack (e.g., Substitute [22]) gives the adversary no access to the DNN information; only leveraging input image and output score of the DNN. Here we briefly introduce the adversarial example generation techniques we used to evaluate our method:

PGD Attack. Projected gradient descent (PGD) [7] is a popular white-box adversarial input attack. It is one of the strongest L_∞ norm based attack that iteratively generates malicious samples $\hat{\boldsymbol{x}}$ from clean (i.e., no noise) samples $\boldsymbol{x}$ with label y. At each iteration t, PGD follows the update rule:

$$\hat{\boldsymbol{x}}^{t+1} = \hat{\boldsymbol{x}}^{t} + \alpha \cdot sign\big(\nabla_{\boldsymbol{x}} \hat{\mathcal{L}}(f(\hat{\boldsymbol{x}}^{t}; \boldsymbol{\theta}), y)\big), \qquad (1)$$

where $f(;)$ is the DNN inference function parameterized by θ, α is the step size, and $\hat{\boldsymbol{x}} \in [0, 1]$ for normalized input. PGD attack [7] generates universal and strong adversary among the first order approach (i.e., attack relying on only first order gradient information) by adding the gradient sign of the loss function $\mathcal{L}$ with regard to the input $\boldsymbol{x}$.

Substitute Model Attack. Prior works [9] have demonstrated that non-linear function of DNNs causes *gradient obfuscation* (i.e., attacker fails to approximate the true gradient), which render the white-box attacks to perform poorly. One possible solution to bypass this obfuscation issue is to evaluate defenses against black-box attacks (e.g., substitute model [22]) that do not require any gradient information. The adversary can train a substitute model known as *source* from the *target* model to exactly mimic the functionality of the target model [22].

Subsequently, an attacker can use the source model to generate a strong adversary using any white-box attack (e.g., PGD) and transfer the adversary to the target model. This attack is also illustrated in Fig. 2.

In a related track, the vulnerability of DNNs against adversarial weight attacks [10], [11], [23] have been actively investigated. Among them, bit flip attack (BFA) [10], [11] has proven to be the most effective, which demonstrated accuracy collapse of ResNet-18 for ImageNet from 69% to 0.1% by modifying only 13 bits out of 88 million bits.

Bit Flip Attack (BFA). BFA integrates progressive search and gradient ranking to identify the vulnerable bits in quantized DNNs. For each attack iteration, BFA follows two steps: i) *In-layer search:* The attacker picks each layer of the DNN and flips top n_b gradient bits (i.e., n_b = 1 typically) to record the inference loss. After evaluating the loss, the attacker restores the original bit state [10]. ii) *Cross-layer search:* In this step, the attacker picks the layer with maximum inference loss evaluated at the last step and performs the bit-flip at that layer. In addition, deep hammer attack [11] has demonstrated that the vulnerable bits identified by BFA can be flipped in real hardware through popular fault injection techniques such as row-hammer [24]. The key advantage of BFA is that quantized networks were attacked successfully (i.e., lowering accuracy to random guess), whereas other works [23] showed unsuccessful weight attack for quantized DNNs.

C. Adversarial Defense with Noise Injection and Quantization

A common approach to address the challenge of adversarial examples is to train DNNs using adversarial samples, which is popularly known as *Adversarial Training* [7]. This optimizes the network with both clean and malicious samples:

$$\arg\min_{\boldsymbol{\theta}} \left\{ \arg\max_{\boldsymbol{x}'} \mathcal{L}\big(f(\hat{\boldsymbol{x}};\boldsymbol{\theta}), y\big) \right\} \tag{2}$$

Here, the inner maximization generates adversarial samples $\hat{\boldsymbol{x}}$ 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.

Several works further improved adversarial training by injecting noise at both training and inference phases [13], [25]. Injecting noise during training works as a regularizer to prevent DNNs from over-fitting [13], [26] and also aids optimization between clean accuracy (i.e., no attack) and perturbed accuracy (i.e., under attack) [13]. However, injecting noise during adversarial training causes gradient obfuscation. Several works, instead, have quantized the DNN weights [27] during training to leverage gradient obfuscation only as a defense tool. On the other hand, aggressive model quantization (i.e., binary weights) [12] is also proven to be largely effective in resisting adversarial weight attack (e.g., BFA), but still cannot completely defend it.

III. Proposed Adversarial Robustness Schemes Using IMC-based Noise and Quantization

In this section, we present the proposed schemes to enhance the robustness of DNNs against adversarial attacks, exploiting the inherent noise/variability of IMC hardware and evaluating partial sum quantization at the IMC crossbar granularity. Our framework incorporates the following aspects:

- PGD based adversarial training [7] (Section II.C) with smooth CELU activation function [20]
- In-training *activation and weight quantization* [28] for low-precision DNNs (e.g. 1-bit, 2-bit, 4-bit)
- Involving IMC noise for DNN inference and training based on actual IMC prototype chip measurements
- *Partial sum quantization* (e.g. 1-bit, 2-bit, ~3-bit) considering IMC crossbar size, ADC, and input-splitting [18]

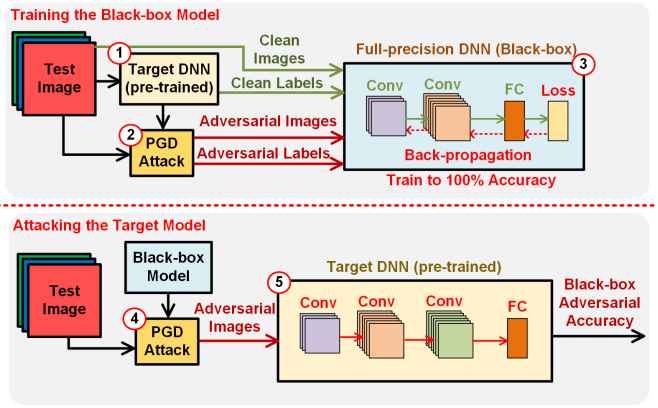


Fig. 2. Illustration of the black-box PGD attack method.

A. Adversarial Training with CELU

Several adversarial attacks require the gradients of DNNs to generate adversarial images. Therefore, various functions used in DNNs must be continuously differentiable. While ReLU is one of the commonly used activation functions, the gradient of ReLU has an abrupt change at input of zero. Such discontinuity lowers the quality of gradients, and weaker adversarial examples would be used for adversarial training of DNNs. To make the gradient continuously differentiable, we employ the CELU activation function [16], [20], which is defined as:

$$CELU(x,\alpha) = \begin{cases} x, & \text{if } x \geq 0 \\ \alpha(exp(\frac{x}{\alpha}) - 1) & \text{otherwise} \end{cases} \tag{3}$$

B. Training DNNs with IMC Quantization and Noise

To train DNNs for inference with very low precision such as 1-bit, 2-bit, and 4-bit, in-training quantization [28] becomes a necessity. In IMC hardware targeting low-precision DNN inference, each IMC crossbar performs MAC operations to obtain the partial sum for a fixed number of inputs (e.g. 256-input partial sum), and the partial sums are quantized to a limited number of ADC levels. Due to the hardware noise and variability (e.g. supply noise, mismatch of transistors, wires, and capacitors), the partial sums that have the same MAC value could result in different ADC outputs.

We employ such hardware noise obtained from IMC prototype chip measurements in two ways. First, we only involve noise for DNN inference for pre-trained 1-bit, 2-bit, and 4-bit DNNs. Second, we inject IMC hardware noise during DNN

training at the partial sum level (as measured from the IMC prototype chip), so that DNNs become aware of the noisy quantization of partial sums and adapt the weights accordingly.

C. Aggressive Quantization of Partial Sums in IMC Crossbars

IMC crossbar supports a fixed number of inputs and weights per dot-product computation and generates intermediate analog partial sums. These partial sums are digitized and accumulated outside the IMC crossbar to represent the final output of the layer, also known as a full sum. IMC hardware typically uses multi-bit ADCs to digitize these partial sums performed by a column of the IMC SRAM array, and additional area and energy costs need to be spent to accommodate such ADCs.

The input-splitting scheme has been presented in [18], [19] to address this issue of large ADCs required in IMC hardware. The Input-splitting algorithm divides the convolution and fully-connected layers into groups, where each group has the same number of inputs as the IMC crossbar (e.g., 256) and computes partial sums. During the DNN training process, the partial sums are aggressively quantized to 1-bit or 2-bit values, and the DNNs are trained to adapt to such computations. This helps reducing the high-resolution ADCs to single comparators or 2-bit ADCs, but we posit that the small adversarial perturbations on inputs or weights of DNNs could be masked by such aggressive partial sum quantization, improving the adversarial robustness. Table I summarizes the thresholds used in the aggressive partial sum quantization scheme in this work.

D. Adversarial Input Attack: Black-box Attack and Evaluation

To circumvent the issue of potential gradient obfuscation [9] present in our low-precision DNNs with IMC noise and partial sum quantization, we used the black-box adversarial attack as illustrated in Fig. 2.

We first pre-train the target DNNs with low-precision and IMC noise (e.g. with gradient obfuscation), and we obtain the predicted labels for the clean images using the pre-trained target model. Then, we train a full-precision black-box DNN (e.g. without gradient obfuscation) using the same input images and corresponding white-box adversarial images obtained from the target model. This black-box model is trained to 100% accuracy with respect to the predicted labels of the target model, and the PGD adversarial attack is applied. Then, the adversarial images generated by the black-box model attack are used to evaluate the adversarial accuracy of the target DNNs with low-precision and IMC noise.

E. Adversarial Weight Attack: Bit-Flip Attack and Evaluation

We adopt the BFA attack delineated in [10] and performed on DNNs implemented with IMC hardware. The un-targeted BFA attack uses progressive search and gradient ranking to identify vulnerable bits that degrade test accuracy. The objective of the attacker is to lower the overall test accuracy by maximizing the loss function:

$$\max \ \mathcal{L} = \max_{\{\hat{W}\}} \ \mathbb{E}_{\mathbb{X}} \mathcal{L}(f(\boldsymbol{x}, \hat{W}); \boldsymbol{t}), \tag{4}$$

TABLE I
MAC THRESHOLDS AND OUTPUT LEVELS FOR AGGRESSIVE PARTIAL SUM QUANTIZATION SCHEMES

ADC Precision	MAC Thresholds	MAC Output Levels
1 bit	0	-1, +1
2 bits	-24, 0, +24	-36, -12, +12, +36
3.5 bits	[-54,+54], step = 12	[-60,+60], step = 12

where $\hat{W}$ is the weight matrix after flipping the target bits, and $f(.)$ is the DNN inference function with loss $\mathcal{L}$. To conduct the attack, we assume the attacker has access to a sample batch of data $\boldsymbol{x}$ and corresponding true label $\boldsymbol{t}$.

To progressively search for vulnerable bits, at each attack iteration, we flip the top n_b ranked bits (e.g., typically n_b=1) based on the gradient of every bit in each of the P layers of the DNN. Similar to [10], we only flip the bits in the direction of its gradient sign. After flipping the bits at a given layer, we evaluate the loss $\mathcal{L}$, and restore the flipped bits to the original state. This way, we generate a loss profile set of $\{\mathcal{L}^1, \mathcal{L}^2, \cdots, \mathcal{L}^P\}$, and identify the layer with maximum loss:

$$j = \arg\max_{l} \ \{\mathcal{L}^l\}_{l=1}^{P} \tag{5}$$

Finally, the attacker enters layer j to perform the bit-flip of the current iteration. The attack iterates until DNN accuracy degrades to a random guess (i.e., 10% for CIFAR-10).

IV. EXPERIMENT

A. Experiment Setup

Against adversarial attacks, we analyzed both conventional DNN training and adversarial training. We primarily used the ResNet-18 DNN as the target model with 1-bit, 2-bit, and 4-bit precision in activations and weights. We performed adversarial input and weight attacks, where we used the PGD algorithm [7] as the main adversarial input attack with ϵ=0.03, α=2/255, and iterations=10, and the BFA [10] as the main adversarial weight attack. All DNNs were trained using either the Adam or the SGD optimization algorithm in the PyTorch framework.

Starting from the in-training quantization scheme [28], we made further modifications in the DNN training and inference process to integrate IMC hardware noise injection and input-splitting (1-bit and 2-bit) quantization of partial sums. We performed adversarial training of DNNs by using both the clean images and corresponding adversarial images obtained using the white-box PGD attack.

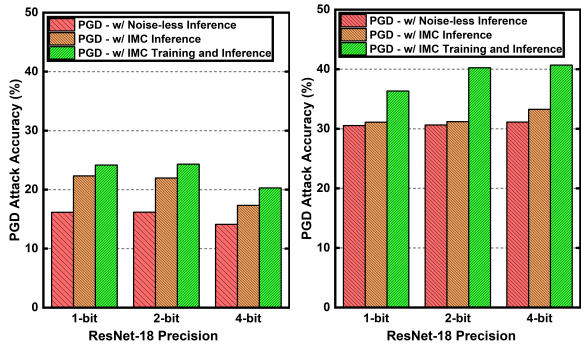
We also trained DNNs with 1-bit and 2-bit partial sum quantization by expanding the previous input-splitting work [18]. The ADC comparator thresholds and levels used for 3.5-bit (11-level) [3], 2-bit, and 1-bit partial sum quantization are shown in Table I. We experimented with different fixed threshold values for DNNs with partial sum quantization, and then tuned the IMC prototype chip [3] using the best threshold values to extract the IMC hardware noise data.

B. Adversarial Input Attack and Defense Results

Table II shows the clean and black-box adversarial accuracies for binary ResNet-18 trained with IMC noise characteristics measured at different supply voltages. Note that

TABLE II
BLACK-BOX PGD ATTACK ACCURACIES FOR BINARY RESNET-18 DNN WITH NOISE-AWARE TRAINING USING DIFFERENT NOISE MODELS

Noise Model (Training and Inference)	Relative Noise Intensity	No Adversarial Training		PGD Adversarial Training	
		No Attack	BB Attack	No Attack	BB Attack
None	1	89.86%	24.20%	86.33%	34.54%
XNOR-SRAM 0.6V [3]	5.99	88.25%	26.97%	86.21%	36.34%
XNOR-SRAM 0.8V [3]	12.92	88.91%	29.20%	85.72%	37.11%
XNOR-SRAM 1.0V [3]	18.63	87.19%	30.12%	83.46%	38.63%
C3SRAM [5]	4.12	88.16%	27.29%	86.03%	35.21%
PNI Noise [13]	1.13	90.12%	26.38%	86.22%	36.21%



(a) Conventional training (b) Adversarial training

Fig. 3. Black-box adversarial accuracy with IMC-noise-aware training based on IMC measurements at 0.6V [3]. 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).

the noise of XNOR-SRAM IMC chip increased with higher supply voltages [3], due to larger IR drop on the bit-lines. With a higher amount of IMC noise, the clean accuracy (no attack) slightly degrades, but the adversarial accuracy (black-box attack) notably improved, since injecting a higher amount of noise during DNN training led to a stronger generalization.

We evaluated the effect of adversarial training and IMC-noise-aware training on black-box adversarial accuracies for ResNet-18 DNNs with 1-bit, 2-bit, and 4-bit activation/weight precision, as shown in Fig. 3. It can be seen that adding IMC noise to inference and training progressively increases the PGD attack accuracy. In comparison to the baseline noise-less model, the accuracy is improved by up to ∼10% by adding measured IMC noise from [3] to the DNN training and inference process. Compared to the conventionally trained DNNs in Fig. 3(a), the DNNs with adversarial training in Fig. 3(b) shows largely improved robustness across all DNNs. The noisy partial sum quantization of IMC noise acts as an inherent regularizer and teaches the DNN to be more tolerant to fluctuations in the partial sum values. Therefore, with IMC-noise-aware training, the DNN becomes more robust against adversarial attacks that perturb the input signal by a small amount.

Considering the partial sum quantization (PSQ) for IMC inference (e.g. 3.5-bit in [3], we experimented the corresponding

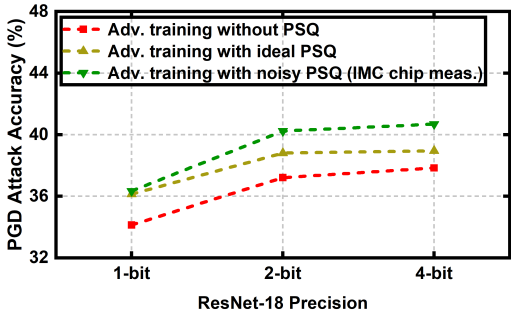


Fig. 4. The effect of ideal vs. noisy partial sum quantization (PSQ) during adversarial training on black-box adversarial accuracy for ResNet-18 DNNs.

3.5-bit PSQ during the adversarial training of DNNs. In Fig. 4, the adversarial accuracy results of ideal 3.5-bit PSQ and IMC chip measurement based noisy 3.5-bit PSQ during adversarial training are shown for 1-bit, 2-bit, and 4-bit ResNet-18 DNNs. It can be seen that adding IMC noise during training gives the best robustness accuracy results. With IMC-noise-aware training, the black-box adversarial accuracy is improved by up to 7% on average across 5 runs of PGD attacks, compared to adversarial training without IMC noise.

We evaluate the effect of aggressive partial sum quantization (i.e., input splitting) in Fig. 5, where the black-box attack accuracies are shown for ResNet-18 DNNs without adversarial training. During training of the same DNNs, we aggressively quantized the 256-input partial sums (fitting the IMC crossbar size) to binary values, which resulted in ∼3% clean accuracy degradation. Aided by input-split DNN training and IMC hardware noise, however, the adversarial attack accuracy improved by up to 4.77%, 4.28%, and 5.74% for 1-bit, 2-bit, and 4-bit ResNet-18 DNNs, respectively.

C. Adversarial Weight Attack and Defense Results

We performed BFA for 1-bit, 2-bit, and 4-bit ResNet DNNs for CIFAR-10, and the results are shown in Fig. 6. Compared to the baseline BFA (no noise), when we applied the XNOR-SRAM IMC noise results from 3.5-bit ADC [3], the DNNs became more vulnerable to BFA (requiring fewer bits to reach ∼10% CIFAR-10 accuracy). However, the input-splitting

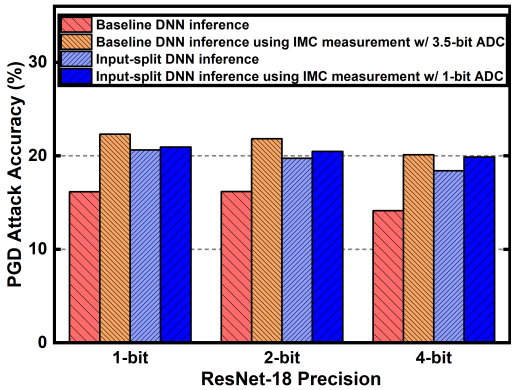
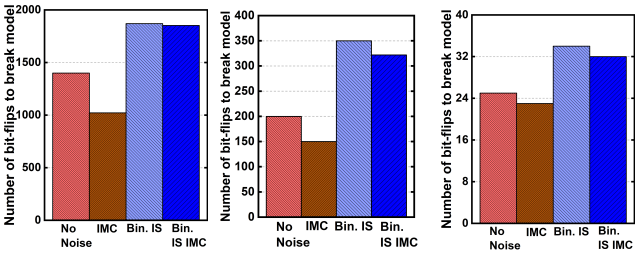


Fig. 5. The effect of input-splitting and IMC noise on black-box adversarial accuracy is shown for ResNet-18 DNNs with 1-bit, 2-bit, and 4-bit weights and activations. The accuracies are shown for DNNs that are not adversarially trained, and for XNOR-SRAM IMC noise measured at 0.6V [3].



(a) 1-bit ResNet-18 (b) 2-bit ResNet-18 (c) 4-bit ResNet-18

Fig. 6. BFA performance of different ResNets with no noise, IMC noise measured using 3.5-bit ADC ("IMC"), binary input-split DNN model ("Bin. IS"), and measured binary input-split DNN with IMC noise ("Bin. IS IMC").

TABLE III
COMPARISON TO PRIOR WORKS

Parameter	[13]	[17]	This Work
IMC type	N/A	NVM simulation	**SRAM chip [3], [5] measurements**
Array size	N/A	64x64	**256x64**
Quantization	Activations Weights	Activation only	**Activations Weights**
Adversarial Training	Yes	No	**Yes**
White-box Adv. Accuracy Improvement	N/A	2.16%	**2.77%**
Black-box Adv. Accuracy Improvement	9.83%	7.80%	**10.52%**

scheme with partial sum binarization required BFA to flip 33.57% more bits to reach random guess, showing enhanced robustness against BFA. When we used IMC chip measurement with partial sum binarization with 1-bit ADC (single comparator), a similar level of robustness was maintained against BFA, overall requiring >30% more bit-flips compared to the baseline BFA. Also, it should be noted that binary DNNs require ∼6X and ∼50X more bit-flips, compared to 2-bit and 4-bit DNNs, respectively (Fig. 6).

D. Comparison to Prior Works

In Table III, the comparison to two relevant prior works is shown. Compared to PNI [13], this work can incorporate arbitrary IMC hardware noise and achieves better black-box adversarial accuracy improvement. [17] evaluated NVM IMC for different array sizes, but only used ideal simulation models and is not based on actual IMC silicon results. By integrating actual IMC prototype chip results [3], [5] in the DNN training/inference process, our scheme shows better adversarial robustness. In addition, ours is the only work that investigated both adversarial input attacks and weight attacks.

V. CONCLUSION

In this work, we reported a new DNN training scheme that integrates measured IMC noise and aggressive partial sum quantization at the IMC crossbar. We show that the proposed scheme effectively improves the robustness of IMC DNN hardware against adversarial input and weight attacks. For PGD input attacks, black-box adversarial accuracy was improved by up to 10%. Against the bit-flip weight attacks, our proposed scheme requires >30% additional bit-flips.

ACKNOWLEDGMENT

This work is partially supported by NSF grants 1652866, 1715443, 2005209, and 2019548, and C-BRIC, one of six centers in JUMP, a SRC program sponsored by DARPA.

REFERENCES

[1] L. Deng *et al.*, "Model Compression and Hardware Acceleration for Neural Networks: A Comprehensive Survey," *Proc. of the IEEE*, vol. 108, no. 4, pp. 485–532, 2020.

[2] X. Si *et al.*, "A Twin-8T SRAM Computation-in-Memory Unit-Macro for Multibit CNN-Based AI Edge Processors," *IEEE JSSC*, vol. 55, 2020.

[3] S. Yin *et al.*, "XNOR-SRAM: In-Memory Computing SRAM Macro for Binary/Ternary Deep Neural Networks," *IEEE JSSC*, vol. 55, no. 6, pp. 1733–1743, 2020.

[4] Q. Dong *et al.*, "A 351TOPS/W and 372.4GOPS Compute-in-Memory SRAM Macro in 7nm FinFET CMOS for Machine-Learning Applications," in *IEEE ISSCC*, 2020.

[5] Z. Jiang *et al.*, "C3SRAM: An In-Memory-Computing SRAM Macro Based on Robust Capacitive Coupling Computing Mechanism," *IEEE JSSC*, vol. 55, no. 7, pp. 1888–1897, 2020.

[6] H. Valavi *et al.*, "A 64-Tile 2.4-Mb In-Memory-Computing CNN Accelerator Employing Charge-Domain Compute," *IEEE JSSC*, vol. 54, no. 6, pp. 1789–1799, 2019.

[7] A. Madry *et al.*, "Towards Deep Learning Models Resistant to Adversarial Attacks," in *ICLR*, 2018.

[8] I. J. Goodfellow *et al.*, "Explaining and Harnessing Adversarial Examples," *arXiv preprint 1412.6572*, 2014.

[9] A. Athalye *et al.*, "Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples," *ICML*, 2018.

[10] A. S. Rakin *et al.*, "Bit-Flip Attack: Crushing Neural Network with Progressive Bit Search," in *IEEE ICCV*, 2019, pp. 1211–1220.

[11] F. Yao *et al.*, "Deephammer: Depleting the intelligence of deep neural networksthrough targeted chain of bit flips," in *USENIX Security Symposium*, 2020.

[12] Z. He *et al.*, "Defending and Harnessing the Bit-Flip based Adversarial Weight Attack," in *IEEE CVPR*, 2020.

[13] Z. He *et al.*, "Parametric noise injection: Trainable randomness to improve deep neural network robustness against adversarial attack," in *IEEE CVPR*, 2019.

[14] X. Liu *et al.*, "Towards Robust Neural Networks via Random Self-ensemble," *arXiv preprint 1712.00673*, 2017.

[15] A. Jeddi *et al.*, "Learn2Perturb: an End-to-end Feature Perturbation Learning to Improve Adversarial Robustness," in *IEEE CVPR*, 2020.

[16] C. Xie *et al.*, "Smooth Adversarial Training," *arXiv:2006.14536*, 2020.

[17] D. Roy *et al.*, "Robustness Hidden in Plain Sight: Can Analog Computing Defend Against Adversarial Attacks?" *arXiv:2008.1201*, 2020.

[18] Y. Kim *et al.*, "Input-Splitting of Large Neural Networks for Power-Efficient Accelerator with Resistive Crossbar Memory Array," in *ACM/IEEE ISLPED*, 2018.

[19] S. Yin *et al.*, "Monolithically Integrated RRAM and CMOS based In-Memory Computing Optimizations for Efficient Deep Learning," *IEEE Micro*, vol. 39, no. 6, pp. 54–63, 2019.

[20] J. T. Barron, "Continuously Differentiable Exponential Linear Units," *arXiv preprint 1704.07483*, 2017.

[21] C. Xue *et al.*, "A 22nm 2Mb ReRAM Compute-in-Memory Macro with 121-28TOPS/W for Multibit MAC Computing for Tiny AI Edge Devices," in *IEEE ISSCC*, 2020.

[22] N. Papernot *et al.*, "Practical Black-Box Attacks Against Machine Learning," in *ACM Asia Conf. on Computer and Comm. Security*, 2017.

[23] S. Hong *et al.*, "Terminal Brain Damage: Exposing the Graceless Degradation in Deep Neural Networks Under Hardware Fault Attacks," in *USENIX Security Symposium*, 2019, pp. 497–514.

[24] Y. Kim *et al.*, "Flipping Bits in Memory without Accessing Them: An Experimental Study of DRAM Disturbance Errors," in *ACM ISCA*, 2014.

[25] M. Lecuyer *et al.*, "Certified Robustness to Adversarial Examples with Differential Privacy," in *IEEE Symp. on Security and Privacy*, 2019.

[26] N. Srivastava *et al.*, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *JMLR*, vol. 15, pp. 1929–1958, 2014.

[27] J. Lin *et al.*, "Defensive Quantization: When Efficiency Meets Robustness," in *ICLR*, 2019.

[28] I. Hubara *et al.*, "Quantized Neural Networks: Training Neural Networks with Low Precision Weights and Activations," *JMLR*, 2017.