Robustness Hidden in Plain Sight: Can Analog Computing Defend Against Adversarial Attacks?

Deboleena Roy  
Department of ECE  
Purdue University  
West Lafayette, IN 47907  
roy77@purdue.edu

Indranil Chakraborty  
Department of ECE  
Purdue University  
West Lafayette, IN 47907  
icharka@purdue.edu

Timur Ibrayev  
Department of ECE  
Purdue University  
West Lafayette, IN 47907  
tibrayev@purdue.edu

Kaushik Roy  
Department of ECE  
Purdue University  
West Lafayette, IN 47907  
kaushik@purdue.edu

Abstract

The ever-increasing computational demand of Deep Learning has propelled research in special-purpose inference accelerators based on emerging non-volatile memory (NVM) technologies. Such NVM crossbars promise fast and energy-efficient in-situ matrix vector multiplications (MVM) thus alleviating the long-standing von Neuman bottleneck in today’s digital hardware. However the analog nature of computing in these NVM crossbars introduces approximations in the MVM operations. In this paper, we study the impact of these non-idealities on the performance of DNNs under adversarial attacks. The non-ideal behavior interferes with the computation of the exact gradient of the model, which is required for adversarial image generation. In a non-adaptive attack, where the attacker is unaware of the analog hardware, we show that analog computing offers a varying degree of intrinsic robustness, with a peak adversarial accuracy improvement of 35.34%, 22.69%, and 31.70% for white box PGD (\(\epsilon=1/255\), iter=30) for CIFAR-10, CIFAR-100, and ImageNet (top-5) respectively. We also demonstrate “hardware-in-loop” adaptive attacks that circumvent this robustness by utilizing the knowledge of the NVM model. To our knowledge, this is the first work that explores the non-idealities of analog computing for adversarial robustness.

1 Introduction

Deep Learning [23] is a popular, versatile machine learning methodology that has been applied to a wide range of optimization tasks, such as computer vision [31], natural language processing [56], recommender systems [10], etc. Many consumer applications rely on deep neural networks (DNNs) to enhance their user experience, such as smart wearables, smart assistants, etc. As our reliance on deep learning increases, so does the need to build secure, reliable, efficient frameworks for executing the intensive computational requirements of DNNs.

To accommodate the growing computational needs of DNNs special-purpose accelerators such as Google TPU [20], Microsoft BrainWave [11], and NVIDIA V100 [11] have been proposed. These systems operate on the principle of efficiently performing matrix-vector multiplication (MVM) operations, the key computational kernel in DNNs, by co-locating memory and processing elements. Despite their success, the saturating scaling trends of digital CMOS [35] has garnered interest in non-volatile memory (NVM) technologies such as RRAM [32], PCRAM [33], and Spintronics [14]. The memory element in these technologies can be arranged in a crossbar fashion to enable efficient
MVM computations in the analog domain inside the memory array. Such an in-memory computing primitive can significantly lower power and latency compared to digital CMOS [29]. Promises offered by the NVM crossbars have propelled significant research in designing analog computing based accelerators, such as PUMA [3].

In an analog computing hardware the output of an MVM operation is sensed as a summation of currents through resistive NVM devices arranged in a crossbar, and hence are prone to errors due to non-ideal behavior of the crossbar and its peripheral circuits. Such errors are hard to model due to the interdependence of multiple analog variables (voltages, currents, resistances) in the crossbar. These deviations result in overall performance degradation of the DNN implementation [8]. Several works have explored various techniques to counteract the impact of these non-idealities [24, 9].

On the flip side, even though the changes in DNN activations arising from non-idealities is hard to model, it can potentially lead to adversarially robust DNN implementations. Adversarial images are generated by estimating the gradients of the model with respect to its input, and carefully perturbing the images in the direction of maximum change in the classifier output [30, 15]. To counter such attacks, several techniques that rely on gradient obfuscation have been previously proposed [13, 5, 4].

In this work, we explore how non-ideal NVM crossbars have a similar intrinsic effect to gradient obfuscation. We implement DNNs on the PUMA architecture, which is composed of thousands of MVM units (MVMUs) made of NVM crossbars. The aforementioned errors occur at the output of these internal MVMUs, which are practically inaccessible to a third party user, such as the software designer or even an attacker. Moreover, the nature of the errors depends heavily on the technology, which might not be fully disclosed by the manufacturer. Finally, any scaled technology is prone to chip to chip variations [26] which can further deter an attacker from exactly replicating the DNN activations. We study two distinct scenarios, one where the attacker does not have access to custom NVM hardware and generates attacks based on accurate digital hardware, and the other where the attacker generates attacks with the NVM hardware in loop.

The main contributions of this work are as follows:

- We demonstrate that adversarial attacks crafted without the knowledge of the hardware implementation are less effective in both black box and white box scenarios.
- We tested multiple variants of NVM crossbars, and show that the degree of intrinsic robustness offered by the analog hardware is in proportion to its degree of non-ideal behavior.
- We show that “Hardware-in-Loop” adaptive adversarial attacks are more effective, as the attacker can now account for the non-ideal computations when crafting the adversarial examples. We show that the degree of success depends on what hardware is available to the attacker and how similar it is to the target model’s hardware.

2 Background and Related Work

![Diagram of NVM crossbar]

Figure 1: (Left) Illustration of NVM crossbar which produces output current $I_j$, as a dot-product of voltage vector, $V_i$ and NVM device conductance, $G_{ij}$. (Right) Various peripheral and parasitic resistances modify the dot-product computations into an interdependent function of the analog variables (voltage, conductance and resistances) in a non-ideal NVM crossbar.
2.1 In-memory Analog Computing Hardware

In-memory analog computing with NVM technologies are being extensively studied for machine learning (ML) workloads [6] because of their inherent ability to perform efficient matrix-vector multiplications, the key computational kernel in DNNs. The basic compute fabric in NVM technologies is a two-dimensional cross-point memory, known as a crossbar, shown in Fig. [1]. The memory devices lie at the intersection of horizontally (source-line) and vertically (bit-line) running metal lines. The conductance of each memory device can be programmed to a discrete number of levels [19]. By simultaneously applying inputs, in the form of voltages, \( V_i \), at the source-lines, the multiplications are performed between the voltages, \( V_i \) and conductances, \( G_{ij} \), by each NVM device using the principle of Ohm’s law. Finally, the product, which is the resulting current, \( I_{ij} \), from each NVM device, is summed up using Kirchhoff’s current law to produce a dot-product output, \( I_j \) at each column:

\[
I_j = \sum_{i} I_{ij} = \sum_{i} V_i G_{ij}
\] (1)

Such parallelized dot-products across all columns enable efficient multiplication of the input voltage vector, \( V \), and the crossbar conductance matrix, \( G \), resulting in an output vector, \( F = VG \). A few key aspects of the design of NVM crossbars are the following parameters:

- Crossbar Size: The number of rows and columns in the crossbar matrix.
- ON Resistance (\( R_{ON} \)): The minimum resistance level of the NVM device.

Typically, in a convolutional neural network (CNN), the convolution operation between the input and the weight tensor can be represented in the form of a series of MVM operations, which can be subdivided into smaller MVM operations to conform to the technological restrictions of the size of the NVM crossbar. Floating point inputs and weights in DNNs are converted to fixed point precision to make them compatible with NVM crossbar based computations.

The analog nature of computing in NVM crossbars introduces functional errors in the MVM computations due to several non-idealities arising from the NVM devices and peripheral resistances. The aforementioned crossbar design parameters, such as Crossbar Size, and ON Resistance have varying impact on the degree of functional errors introduced by the non-idealities [8] by affecting the effective resistance of a crossbar column. Larger crossbar size lowers the effective resistance, making the crossbar more prone to non-ideal effects, while higher ON resistance increases it, resulting in a crossbar less affected by non-idealities.

Due to the non-idealities, the resulting output current, \( I_{ni} \) is a function of voltage vector \( V \), conductance matrix \( G(V) \), which is now dependent on \( V \), and several non-ideal factors:

\[
I_{ni} = f(V, G(V), R_{source}, R_{sink}, R_{wire})
\] (2)

To study the impact of such non-ideal behavior of NVM crossbars on DNNs, researchers have previously proposed techniques to model the non-ideal function in Equation [2]. One such technique is GENIEx [8] where the authors use a neural network to model the aforementioned non-ideal function.

DNNs typically consist of thousands of MVM operations at every layer. The NVM crossbar non-idealities cause the activations at every layer to deviate from their expected value, and this deviation propagates through the network. This results in a degradation of DNN accuracy at inference (without any adversary). Interestingly, the same deviation in activation imparts adversarial robustness when under attack, which is further analyzed in this paper.

2.2 Adversarial Attacks

In 2013, the authors of [30] demonstrated that a classifier can be forced to make an error by adding small perturbations to the data which are almost imperceptible to the human eye. They coined the term "adversarial examples" to define such data designed specifically to fool the classifier. Since then, several methods have been developed to generate such data, which are known as "adversarial attacks". In principle, these attacks try to solve the following optimization problem [27]:

\[
x^* = x + \arg\min_{z} \{ z : F(x + z) \neq F(\theta, x) \} = x + \delta_x
\] (3)

where \( x \) is the original data, \( x^* \) is the perturbed adversarial data, \( F(x) \) is the classifier function, mapping inputs to labels, and the objective of the adversary is to misclassify, i.e. \( F(x^*) \neq F(x) \). Most attacks use gradient-based optimization to solve for eq[3] and the attack’s success relies on how accurately one can estimate \( \nabla_x L(\theta, x, y) \), the derivative of the cost function \( L(\theta, x, y) \) with respect to \( x \), where \( \theta \) is the target model parameters, and the inputs and labels are \( x \) and \( y \) respectively [15].
3  Adversarial Robustness of NVM Crossbar based Analog Computing

In recent years, several adversarial defenses have been proposed that disrupt the gradient computation of the model by adding an extra computational element to the network, such as a randomization layer at the beginning [34], or adaptive dropout after every layer [13]. When a DNN model is implemented on an NVM crossbar architecture, the non-idealities have a similar effect of changing the layer-wise activations of the DNNs. There is no simple differentiable function to model these deviations, and one cannot determine them without probing the analog hardware. Thus, such an implementation, could potentially increase the robustness of the neural network. In this section we describe the methodology to emulate DNNs on the PUMA architecture, and set up different threat models based on the attacker’s knowledge of both the software and the hardware.

3.1 Crossbar Models

To model the non-ideal crossbar, we use GENIEx, a deep learning based crossbar model developed by the authors of [8]. They define a multi-layer perceptron (MLP) which receives $V$ and $G$ as inputs and predicts the output $I_{ni}$. This MLP is trained on training pairs $[(V,G), I_{ni}]$ obtained from circuit simulations. In this paper, we have replicated the modeling technique of GENIEx to generate 3 RRAM based crossbar models (Table 1).

The degree of non-ideality has been described by the authors of GENIEx as Non-ideality Factor $(NF) = (\text{Expected output} - \text{Actual Output})/\text{Expected Output}$. $NF$ is directly ( inversely) proportional to crossbar size (ON Resistance). In our experiments, we have considered different crossbar models to study the impact of different degrees of non-idealities, represented by different $NF$, on adversarial robustness, as shown in Table 1.

To integrate the NVM crossbar models with the PyTorch framework, we have adopted a functional simulator from [8] based on PUMA hardware architecture [3].

3.2 Datasets and Network Models

For our evaluation we selected 3 image recognition tasks, and trained a ResNet [18] for each task.

- **CIFAR-10** [21]: A ResNet-20 was trained for 200 epochs, with the learning rate ($lr$) schedule $[0.1(1, 79), 0.01(80, 119), 0.001(120, 200)]$ and achieved test accuracy of 92.44%
- **CIFAR-100** [21]: A ResNet-32 was trained for 200 epochs, with the $lr$ schedule $[0.1(1, 79), 0.01(80, 119), 0.001(120, 200)]$, and achieved test accuracy of 71.42%
- **ImageNet** [12]: A ResNet-18 was trained for 90 epochs, with the $lr$ schedule $[0.1(1, 29), 0.01(30, 59), 0.001(60, 90)]$, and achieved top-1 and top-5 test accuracy of 69.83% and 89.19% respectively.

3.3 Generating Adversarial attacks

To model the non-ideal crossbar, we use GENIEx, a deep learning based crossbar model developed by the authors of [8]. They define a multi-layer perceptron (MLP) which receives $V$ and $G$ as inputs and predicts the output $I_{ni}$. This MLP is trained on training pairs $[(V,G), I_{ni}]$ obtained from circuit simulations. In this paper, we have replicated the modeling technique of GENIEx to generate 3 RRAM based crossbar models (Table 1).

The degree of non-ideality has been described by the authors of GENIEx as Non-ideality Factor $(NF) = (\text{Expected output} - \text{Actual Output})/\text{Expected Output}$. $NF$ is directly ( inversely) proportional to crossbar size (ON Resistance). In our experiments, we have considered different crossbar models to study the impact of different degrees of non-idealities, represented by different $NF$, on adversarial robustness, as shown in Table 1.

To integrate the NVM crossbar models with the PyTorch framework, we have adopted a functional simulator from [8] based on PUMA hardware architecture [3].

3.2 Datasets and Network Models

For our evaluation we selected 3 image recognition tasks, and trained a ResNet [18] for each task.

- **CIFAR-10** [21]: A ResNet-20 was trained for 200 epochs, with the learning rate ($lr$) schedule $[0.1(1, 79), 0.01(80, 119), 0.001(120, 200)]$ and achieved test accuracy of 92.44%
- **CIFAR-100** [21]: A ResNet-32 was trained for 200 epochs, with the $lr$ schedule $[0.1(1, 79), 0.01(80, 119), 0.001(120, 200)]$, and achieved test accuracy of 71.42%
- **ImageNet** [12]: A ResNet-18 was trained for 90 epochs, with the $lr$ schedule $[0.1(1, 29), 0.01(30, 59), 0.001(60, 90)]$, and achieved top-1 and top-5 test accuracy of 69.83% and 89.19% respectively.

3.3 Generating Adversarial attacks
We define 5 different threat scenarios with varying extent of the attacker’s knowledge of the target model and the underlying hardware (Table 1). For each threat scenario there is an attack model (a single DNN or an ensemble of DNNs) which is used to generate the adversarial images. We use Projected Gradient Descent (PGD) [25] to generate iterative perturbations that are bound by the $l_\infty$ norm, as shown in Eq 4.

$$x^{t+1} = \Pi_{x+S}(x^t + \alpha \text{sgn}(\nabla_x L(\theta, x^t, y)))$$

$x^{t+1}$ is the adversarial example generated at $(t+1)^{th}$ iteration. The model’s cost function is $L(\theta, x, y)$, which is a function of the model parameters $\theta$, input $x$, and labels $y$. The set of allowed perturbations is given by $S$. For the $l_\infty$ norm, the attack epsilon ($\epsilon$) defines the set of perturbations as $S = \left\{ (x + \delta \geq \max(x + \epsilon, 0)) \wedge (x + \delta \leq \min(x + \epsilon, 1)) \right\}$, where $x \in [0, 1]$.

3.3.1 Non-Adaptive Attacks

Our first category of threat scenario is "Non-Adaptive Attacks", i.e. the attacker has no knowledge of the underlying analog hardware and the attacks are generated under the assumption of accurate digital computation. Under this category, we have 3 varying degrees of attack.

Transfer Attacks This is the weakest threat model where the adversary has no knowledge of the model. The attack model is another DNN trained on the same dataset and run on an accurate digital hardware. The attack model architectures for CIFAR-10/100, and Imagenet are ResNet-10, ResNet-20, and AlexNet [22], respectively. ResNet-10 and ResNet-20 are trained on CIFAR-10/100, respectively, using the same training schedule as the target models. For Imagenet, we used a pretrained AlexNet available in Pytorch [28].

Black Box Attacks The attacker queries the model on an accurate digital hardware and reads the output of the final layer before softmax (logits) to generate a synthetic dataset of training data and its corresponding logits. This synthetic dataset is used to train 3 different ResNet models, ResNet-10,20,32, which are used to generate adversarial images using the stack parallel ensemble strategy [17].

White Box Attacks This is the highest threat level where the attacker has full knowledge of the model weight, thus the attack model is the same as the target model. However, while generating gradients, the attacker has no knowledge of the underlying analog hardware implementation. The gradients for the attack are computed assuming accurate digital hardware implementation.

Comparison with Related Work We have selected 3 defenses that can be applied to a pretrained network as listed below. For a fair comparison, we apply non-adaptive attacks for these defenses as well, i.e. the defenses are not visible to the attacker when they query the model to generate their synthetic dataset for Black Box attacks, and when they generate gradients for White Box attacks.

- **Input Bit Width (BW) Reduction** [16]: The input is quantized to 4-bits.
- **Stochastic Activation Pruning (SAP)** [13] (for CIFAR-10/100 only): At inference, after every convolution layer, there is an adaptive dropout, that randomly sets the layer outputs to 0 with a probability proportional to their absolute value.
- **Random Padding** [4] (for ImageNet only): Two randomization layers are introduced before the pretrained model. The first layer scales the input image to a random size NxN where $N \in [299, 331]$ using nearest-neighbor extrapolation. The second layer randomly pads the image to generate the final image of size 331x331.

3.3.2 Hardware in Loop Adaptive Attacks

In this category of attacks, the attacker is aware that the model is implemented on an NVM crossbar hardware. However, the crossbar model available to the attacker may or may not match with the target’s implementation. For crafting Black Box Attacks, the attacker queries the DNN model implemented on the NVM crossbar based hardware to create the synthetic dataset. In the case of White Box attacks, the attacker generates adversarial images using "Hardware-in-Loop" gradient descent. Note that the NVM crossbar based hardware is designed for inference tasks and does not support backpropagation of gradients. Thus, for "Hardware-in-Loop", the forward pass is performed on NVM crossbar hardware, and all activations are recorded. However, the derivatives are calculated assuming ideal computations in place of non-ideal MVM operations of the crossbar. As described in Section 3.1, the NVM crossbar non-idealities vary with crossbar properties. We use 3 different crossbar models as defined in Table 1 and we explore scenarios where there is a mismatch in the crossbar model used by the attacker and the target implementation.
4 Results

The first effect of implementing DNNs on a NVM crossbar hardware is the reduction in clean accuracy due to the errors associated with non-ideal computations. Greater the Non-Ideality Factor (NF), more severe is the accuracy degradation as noted in Table 3 and 4. The clean accuracy of CIFAR-10 drops from 92.44% (accurate digital hardware) to 88.34% on 64x64_100k, the most non-ideal crossbar model among the three chosen. Similarly, CIFAR-100 accuracy drops from 71.42% to 55.48% and ImageNet accuracy falls from 69.56% to 62.50% on the 64x64_100k NVM crossbar hardware. If non-idealities of NVM hardware had no impact on adversarial robustness, similar degradation would have been observed in model accuracy under attack. However, our findings, as outlined below, indicate a different trend.

4.1 Non-Adaptive Attacks

Figure 2: Non-Adaptive Transfer Attacks (PGD, iter=30) on CIFAR-10/100 and ImageNet on 3 NVM models and 3 defenses, Input BW Reduction (4-bit input) [16], SAP [13], Random Pad [34]

Transfer Attacks In Fig. 2, we observe the decline in adversarial accuracy with increasing attack epsilon \( \epsilon \) for CIFAR-10, CIFAR-100, and Imagenet. For CIFAR-10/100, the 64x64_300k model did not exhibit any increase in robustness, instead it trailed behind the baseline accuracy. In case of CIFAR-10, the other two crossbar models, 32x32_100k and 64x64_100k, displayed an absolute increase in robustness of 4.2% and 5.9% averaged over \( \epsilon = (2,4,6,8)/255 \), respectively. For CIFAR-100, the average increase in robustness for \( \epsilon = (4,6,8)/255 \) was 1.4% for 32x32_100k and 1.84% for 64x64_100k. The peak improvement in robustness was observed for \( \epsilon = 6/255 \) and has been summarized in Table 3. For ImageNet, we do not observe any improvement in robustness. A possible reason could be that the attack is much weaker, as it was generated on a different architecture (AlexNet), instead of a ResNet. The more generic the attack, the less effect the NVM non-idealities seem to have on robustness.

Figure 3: Non-Adaptive Black Box Attacks (PGD, iter=30) on CIFAR-10, CIFAR-100 on 3 NVM crossbar models and the 2 defenses, Input BW Reduction (4-bit input) [16] and SAP [13]

Black Box Attacks From Fig. 5 we observe similar trends as transfer attacks for CIFAR-10, and CIFAR-100. The 64x64_300k model didn’t exhibit any increase in robustness, instead it trailed behind the baseline accuracy. The NVM crossbar models, 32x32_100k and 64x64_100k, recorded an absolute increase in robustness of 5.3% and 7.8% averaged over \( \epsilon = (2,4,6,8)/255 \), respectively for CIFAR-10. For CIFAR-100, it was 1.4% and 1.84% respectively. The peak improvement in robustness was observed for \( \epsilon = 4/255 \) and has been summarized in Table 3.

White Box Attacks Under this threat model we observe the highest improvement in robustness as depicted in Fig. 4 for CIFAR-10/100 and Table 4 for ImageNet. The NVM model 64x64_300k...
Table 3: CIFAR-10/100 accuracy against Non-Adaptive Attacks (PGD, iter = 30)

| Attack Type                       | Baseline | 64×64_300k | 32×32_100k | 64×64_100k | 4-bit input | SAP       |
|-----------------------------------|----------|------------|------------|------------|-------------|-----------|
| CIFAR-10 (ResNet-20)              |          |            |            |            |             |           |
| Clean                             | 92.44    | 90.35 (-2.09) | 90.42 (+2.02) | 88.34 (-4.10) | 89.84 (+2.60) | 79.76 (-12.68) |
| Transfer Attack (ResNet-10) $\epsilon =$ 6/255 | 12.94    | 12.24 (-0.70) | 18.53 (+5.59) | 21.54 (+8.6) | 22.43 (+9.49) | 30.48 (+17.54) |
| Ensemble Black Box Attack $\epsilon =$ 4/255 | 18.91    | 17.15 (-1.76) | 26.6 (+7.69) | 30.35 (+11.44) | 31.89 (+12.98) | 40.19 (+21.28) |
| White Box Attack $\epsilon =$1/255 | 19.64    | 17.56 (-2.08) | 46.12 (+26.48) | 54.98 (+35.34) | 55.29 (+35.65) | 64.26 (+44.62) |
| White Box Attack $\epsilon =$2/255 | 0.51     | 0.45 (+0.06) | 8.51 (+8.00) | 17.22 (+16.71) | 14.94 (+14.34) | 44.85 (+44.34) |
| CIFAR-100 (ResNet-32)             |          |            |            |            |             |           |
| Clean                             | 71.42    | 63.89 (-7.53) | 62.44 (-9.8) | 55.48 (-15.94) | 64.20 (-7.72) | 44.41 (-27.01) |
| Transfer Attack (ResNet-20) $\epsilon =$ 6/255 | 9.61     | 8.45 (-1.16) | 11.14 (+1.53) | 11.83 (+2.22) | 14.88 (+5.27) | 15.76 (+6.15) |
| Ensemble Black Box Attack $\epsilon =$ 4/255 | 9.88     | 8.03 (-1.85) | 11.95 (+2.07) | 12.59 (+2.71) | 17.07 (+7.19) | 17.60 (+7.72) |
| White Box Attack $\epsilon =$1/255 | 5.78     | 6.53 (+0.75) | 24.22 (+18.44) | 28.47 (+22.69) | 30.45 (+24.67) | 32.4 (+26.62) |
| White Box Attack $\epsilon =$2/255 | 0.24     | 0.39 (+0.15) | 4.55 (+4.31) | 8.27 (+8.03) | 8.94 (+8.70) | 20.14 (+19.59) |

Table 4: ImageNet Accuracy against Non-Adaptive White Box Attacks (PGD, iter = 30)

| Attack Type                       | Baseline | 64×64_300k | 32×32_100k | 64×64_100k | 4-bit input | Random Pad |
|-----------------------------------|----------|------------|------------|------------|-------------|------------|
| Top-1 Accuracy                    |          |            |            |            |             |            |
| CIFAR-10                          | 69.56    | 65.2 (-4.36) | 64.9 (-4.66) | 62.5 (-7.06) | 67.1 (-2.46) | 65.1 (-4.46) |
| White Box Attack $\epsilon =$1/255 | 0.40     | 0.60 (+0.2) | 4.50 (+4.1) | 10.30 (+9.9) | 9.6 (+9.2) | 44.3 (+43.9) |
| White Box Attack $\epsilon =$2/255 | 0.10     | 0.10 (+0.0) | 0.20 (+0.1) | 0.50 (+0.4) | 0.10 (+0.0) | 33.50 (+33.4) |
| Top-5 Accuracy                    |          |            |            |            |             |            |
| CIFAR-10                          | 89.06    | 86.3 (-2.76) | 86.2 (-2.86) | 84.8 (-4.26) | 86.5 (-2.56) | 85.9 (-3.16) |
| White Box Attack $\epsilon =$1/255 | 18.60    | 19.3 (+0.7) | 42.20 (+23.6) | 50.30 (+31.7) | 52.00 (+33.4) | 73.2 (+54.6) |
| White Box Attack $\epsilon =$2/255 | 4.10     | 3.70 (+0.4) | 13.40 (+9.3) | 19.3 (+15.2) | 20.60 (+16.5) | 64.10 (+60.0) |

still continues to closely follow baseline accuracy. For all 3 datasets, the baseline accuracy drops sharply to 0 beyond $\epsilon = 2/255$. At this level, the NVM models are no longer able to recover any performance. For $\epsilon = (1,2)/255$, we observe that 64x64_100k, the most non-ideal of the 3 models, offers the highest improvement for all 3 datasets, with absolute increase of 35.34% for CIFAR-10, 22.69% for CIFAR-100, and 9.90% (32.70%) for ImageNet top-1 (top-5) at $\epsilon = 1/255$.

We observe the following trends for all 3 non-adaptive attacks:

- More the attacker relies on estimating the target model for attack generation, greater is the benefit in robustness. We observed an increase in the absolute improvement from baseline accuracy as we move from Transfer attacks to Black Box to White Box attacks.
- The resulting accuracy is a combination of two opposing forces. The errors caused by the non-idealities try to lower the accuracy, while the intrinsic robustness arising from the same non-idealities lower the effectiveness of the attack and pushes the accuracy higher than the baseline. For example, for 64x64_300k (NF = 0.07), the MVM operations are close to ideal computation for the non-adaptive attacks to transfer successfully. Whereas, the more non-ideal crossbar models, 32x32_100k and 64x64_100k, have greater clean accuracy degradation due to functional errors, but have higher adversarial accuracy, as the non-idealities hinder the transfer of the attacks.
Overall, the intrinsic robustness of NVM crossbars is often within the ball park of Input BW Reduction. However, stronger adversarial defenses such as SAP [13] and Random Padding [34] have performed much better.

Figure 5: Hardware-in-Loop Adaptive Black Box Attacks (PGD, iter=30) on CIFAR-10/100. Target NVM model is 64x64_100k, and the attacks are generated using 3 different NVM models.

Table 5: Hardware-in-Loop Adaptive White Box Attacks (PGD, $\epsilon=1/255$, iter=30)

| Dataset (Attack $\epsilon$) | Baseline 64x64_300k | 32x32_100k | 64x64_100k |
|-----------------------------|----------------------|-------------|-------------|
| Attacker’s NVM Crossbar model: 64x64_100k | | | |
| CIFAR-10                  | 19.64 | 43.45 (+23.81) | 31.78 (+12.14) | 28.84 (+9.2) |
| CIFAR-100                 | 5.78 | 28.21 (+22.43) | 10.86 (+5.08) | 9.73 (+3.95) |
| ImageNet Top-1            | 0.40 | – | – | **0.80** (+0.40) |
| ImageNet Top-5            | 18.60 | – | – | **20.70** (+2.10) |

4.2 Hardware-in-Loop Adaptive Attacks

Black Box Attacks When the attacker builds their synthetic dataset by querying the NVM crossbar hardware implementation of the DNN, the resulting ensemble Black Box attacks are much more effective. The adversarial accuracy of the hardware falls significantly below the baseline, as shown in Fig. 5. Even when the attack is built using a crossbar model different from the target, accuracy degradation is significant. We observe that attacks generated using 32x32_100k (NF = 0.14) are stronger than those generated using 64x64_300k (NF = 0.07) when applied to 64x64_100k (NF= 0.26). This implies that the lesser the difference in NF, the more effective are the attacks.

White Box Attacks The results for hardware in loop White Box attacks are presented in Table 5. The values in bold indicate that attacker’s NVM crossbar model is an exact match to the target model’s underlying hardware. Even when the attacker has full knowledge of the hardware, the non-idealities help improve robustness. We observe that if the attacker’s NVM model is different from the target, the attacks do not transfer well and are weaker than non-adaptive attacks. For example, for CIFAR-10, under attack epsilon $\epsilon = 1/255$, the accuracy of 64x64_300k NVM model is 0.60% for a non-adaptive attack, but 43.45% for an adaptive attack with incorrect NVM model. Thus having an incorrect crossbar model is worse than having no model at all in this case.

5 Discussion

Non-idealities in NVM crossbars have been a long-standing challenge [8] affecting the feasibility of analog computing hardware, and several techniques have been proposed to compensate for it [9]. In this work, we study these non-idealities from the new perspective of adversarial robustness. We observed that DNNs implemented on an NVM crossbar hardware exhibit increased adversarial robustness under varied threat models. While this robustness falls short of other defenses [16, 13, 34], an important point to note is that such robustness is intrinsic to the NVM crossbar hardware, unlike other defenses which have a computational overhead. Also, any algorithmic defense can be further implemented on the analog hardware for additional robustness. The non-ideality factor (NF) of the crossbar model determines the degree of robustness, therefore, one can potentially design NVM crossbars with optimal trade-off between accuracy degradation and increased robustness due to non-idealities. We have demonstrated "Hardware-in-Loop" attacks where the knowledge of underlying hardware helps generate stronger attacks. While we have considered NVM crossbar models based
on RRAM technology [32], analog hardware based on other technologies [33,14] are also possible. This, along with chip to chip variations, may further hinder the transferability of attacks generated on one analog computing hardware to another. In summary, this work is the first step toward understanding the role of non-idealities in NVM crossbar hardware for adversarial robustness. It opens the possibilities of defenses that leverage the non-ideal computations, and on the other hand, attacks that exploit these non-idealities.

**Acknowledgment**

This work was supported in part by the Center for Brain Inspired Computing (C-BRIC), one of the six centers in JUMP, a Semiconductor Research Corporation (SRC) program sponsored by DARPA, the National Science Foundation, Intel Corporation, the DoD Vannevar Bush Fellowship, and by the U.S. Army Research Laboratory and the U.K. Ministry of Defense under Agreement Number W911NF-16-3-0001.

**Broader Impact**

Over the years, deep learning (DL) has shown incredible promise in designing better consumer services, such as smart assistants, virtual health screening, semi-autonomous vehicles, home security, etc. Building reliable, secure and low cost implementations of these methods is important for increasing the outreach of such technological advancements. In recent years, adversarial attacks have exposed the inherent vulnerabilities of deep learning based models and have raised questions about their reliability, especially in mission-critical applications. On the other hand, as deep learning models are scaled to tackle increasingly complex challenges, their computational needs also continue to grow. In such a scenario, a low cost hardware, that is fast, reliable and secure, can support widespread adoption of DL based solutions. In this work, we study how the intrinsic properties of such a low cost hardware contribute to the adversarial robustness of DL models. One can possibly design secure and low cost AI systems of the future by leveraging the interplay between the hardware and algorithm, as demonstrated in this work.

**Acknowledgments and Disclosure of Funding**

The research was funded in part by C-BRIC, one of six centers in JUMP, a Semiconductor Research Corporation (SRC) program sponsored by DARPA, the National Science Foundation, Sandia National Laboratories, Intel Corporation and Vannevar Bush Faculty Fellowship.

**References**

[1] Nvidia tesla v100 gpu architecture, the world’s most advanced data center gpu. Technical report, NVIDIA Corporation, 2017.

[2] Stefano Ambrogio et al. Equivalent-accuracy accelerated neural-network training using analogue memory. *Nature*, 558(7708):60, 2018.

[3] Aayush Ankit, Izzat El Hajj, Sai Rahul Chalamalasetti, Geoffrey Ndu, Martin Foltin, R Stanley Williams, Paolo Faraboschi, Wen-mei W Hwu, John Paul Strachan, Kaushik Roy, et al. Puma: A programmable ultra-efficient memristor-based accelerator for machine learning inference. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 715–731, 2019.

[4] Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *arXiv preprint arXiv:1802.00420*, 2018.

[5] Jacob Buckman, Aurko Roy, Colin Raffel, and Ian Goodfellow. Thermometer encoding: One hot way to resist adversarial examples. In *International Conference on Learning Representations*, 2018.

[6] Geoffrey W Burr et al. Experimental demonstration and tolerancing of a large-scale neural network (165 000 synapses) using phase-change memory as the synaptic weight element. *IEEE Transactions on Electron Devices*, 62(11):3498–3507, 2015.

[7] Fuxi Cai et al. A fully integrated reprogrammable memristor–cmos system for efficient multiply–accumulate operations. *Nature Electronics*, 2(7):290–299, 2019.
[8] Indranil Chakraborty, Mustafa Fayez Ali, Dong Eun Kim, Aayush Ankit, and Kaushik Roy. Geniex: A generalized approach to emulating non-ideality in memristive xbars using neural networks. *arXiv preprint arXiv:2003.06902*, 2020.

[9] Indranil Chakraborty, Deboleena Roy, and Kaushik Roy. Technology aware training in memristive neuromorphic systems for nonideal synaptic crossbars. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2(5):335–344, 2018.

[10] Heng-Tze Cheng, Levent Koc, Jeremiah Harnsme, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*, pages 7–10, 2016.

[11] Eric Chung et al. Serving dnns in real time at datacenter scale with project brainwave. *IEEE Micro*, 38(2):8–20, 2018.

[12] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[13] Guneet S Dhillon, Kamyar Azizzadenesheli, Zachary C Lipton, Jeremy Bernstein, Jean Kossaifi, Aran Khanna, and Anima Anandkumar. Stochastic activation pruning for robust adversarial defense. *arXiv preprint arXiv:1803.01442*, 2018.

[14] Xuanyao Fong, Yusung Kim, Karthik Yogendra, Deliang Fan, Abhironl Sengupta, Anand Raghunathan, and Kaushik Roy. Spin-transfer torque devices for logic and memory: Prospects and perspectives. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 35(1):1–22, 2015.

[15] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[16] Chuan Guo, Mayank Rana, Moustapha Cisse, and Laurens Van Der Maaten. Countering adversarial images using input transformations. *arXiv preprint arXiv:1711.00117*, 2017.

[17] Jie Hang, Keji Han, Hui Chen, and Yun Li. Ensemble adversarial black-box attacks against deep learning systems. *Pattern Recognition*, 101:107184, 2020.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[19] Miao Hu et al. Dot-product engine for neuromorphic computing: programming 1t1m crossbar to accelerate machine-vector multiplication. In *Design Automation Conference (DAC), 2016 53nd ACM/EDAC/IEEE*, pages 1–6. IEEE, 2016.

[20] Norman P Jouppi et al. In-datacenter performance analysis of a tensor processing unit. In *2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA)*, pages 1–12. IEEE, 2017.

[21] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[22] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[23] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[24] Chencheng Liu, Miao Hu, John Paul Strachan, and Hai Li. Rescuing memristor-based neuromorphic design with high defects. In *2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2017.

[25] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[26] Dimin Niu, Yiran Chen, Cong Xu, and Yuan Xie. Impact of process variations on emerging memristor. In *Proceedings of the 47th Design Automation Conference*, pages 877–882, 2010.

[27] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, pages 506–519, 2017.

[28] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alche-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[29] Ali Shafiee et al. Isaac: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars. *ACM SIGARCH Computer Architecture News*, 44(3):14–26, 2016.

[30] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.
[31] Athanasios Voulodimos, Nikolaos Doulamis, Anastasios Doulamis, and Efthychios Protopopadakis. Deep learning for computer vision: A brief review. *Computational intelligence and neuroscience*, 2018, 2018.

[32] H-S Philip Wong, Heng-Yuan Lee, Shimeng Yu, Yu-Sheng Chen, Yi Wu, Pang-Shiu Chen, Byoungil Lee, Frederick T Chen, and Ming-Jinn Tsai. Metal–oxide rram. *Proceedings of the IEEE*, 100(6):1951–1970, 2012.

[33] H-S Philip Wong, Simone Raoux, SangBum Kim, Jiale Liang, John P Reifenberg, Bipin Rajendran, Mehdi Asheghi, and Kenneth E Goodson. Phase change memory. *Proceedings of the IEEE*, 98(12):2201–2227, 2010.

[34] Cihang Xie, Jianyu Wang, Zhishuai Zhang, Zhou Ren, and Alan Yuille. Mitigating adversarial effects through randomization. *arXiv preprint arXiv:1711.01991*, 2017.

[35] Xiaowei Xu et al. Scaling for edge inference of deep neural networks. *Nature Electronics*, 1(4):216–222, 2018.

[36] T. Young, D. Hazarika, S. Poria, and E. Cambria. Recent trends in deep learning based natural language processing [review article]. *IEEE Computational Intelligence Magazine*, 13(3):55–75, 2018.