Robustness for Embedded Machine Learning Using In-Memory Computing

Priyadarshini Panda, Abhiroop Bhattacharjee, and Abhishek Moitra

1 Introduction

Deep learning has achieved state-of-the-art prediction capabilities across a variety of cognitive and analytics tasks. This has led to the ubiquitous deployment of Deep Neural Networks (DNNs) in low power edge devices [1, 2]. For edge computing, analog crossbar architectures have emerged as a front runner towards low-latency and energy-efficient acceleration platforms in resource-constrained scenarios. Here, the synaptic weights of the DNNs are mapped on arrays (crossbars) of Non-Volatile Memory (NVM) devices, such as Resistive RAM (ReRAM), Phase Change Memory (PCM), Ferroelectric Field Effect Transistors (FeFET) and so forth [3, 4]. They efficiently perform analog dot-product operations, emulating Multiply and Accumulate (MAC) operations in DNNs, when input voltages are applied to the rows of the crossbar.

Despite achieving super-human performance in many computer vision tasks [5], DNNs have been shown to be vulnerable to adversarial attacks (see Fig. 1). Here, small and strategically crafted noise in the input can fool the DNN leading to failure [6–10]. This vulnerability severely limits the deployment and potential safe-use of DNNs at the edge for real-world applications. To defend against adversarial attacks, previous works have used two broad approaches: (1) Adversarial classification [11–15] and (2) Adversarial detection [16–18]. Under adversarial classification, there have been prior works that have used techniques such as adversarial training, input feature transformation among others to classify the adversarial samples accurately [11–15]. In contrast, adversarial detection works focus on identifying clean and adversarial samples such that the detected adversarial samples are not passed to

P. Panda (✉) · A. Bhattacharjee · A. Moitra
Department of Electrical Engineering, Yale University, New Haven, CT, USA
e-mail: priya.panda@yale.edu

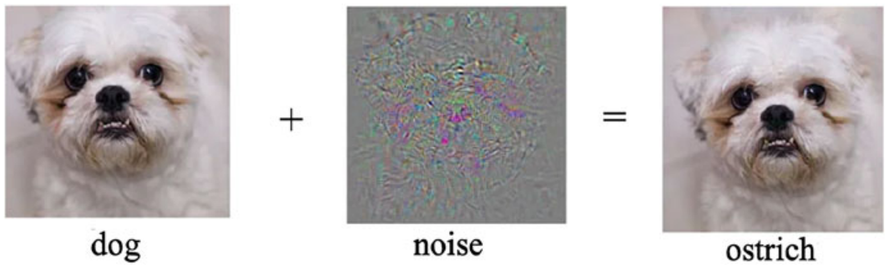


Fig. 1 Adversarial attacks can fool a DNN by adding structured perturbations to clean inputs

the output of the DNN for classification [16–18]. However, these techniques are software-centric and not hardware-friendly, requiring high computational overheads. To this end, recent works such as [10, 19] show that quantization methods, which primarily reduce compute resource requirements of DNNs, act as a straightforward way of improving the adversarial robustness of DNNs. Other works such as [20, 21] use model compression and pruning techniques to optimize and reduce computational complexities of DNNs while guaranteeing adversarial robustness. In [19, 22], efficiency-driven hardware optimization techniques are leveraged to improve adversarial resilience of DNNs, while yielding energy-efficiency. However, none of these works has been integrated with a crossbar-based platform (considering intrinsic crossbar noise) for DNN inference. Vanilla implementation of DNNs on crossbars, including those trained with software defenses such as adversarial training, suffers from significant loss in robustness caused by hardware noise [23–25]. There has been limited study in understanding the robustness of crossbar implemented DNNs. Thus, we highlight hardware and energy-efficiency driven works that improve the robustness of DNNs deployed on analog crossbars in two broad aspects: (1) Improving adversarial robustness and (2) Mitigating the detrimental effects of crossbar non-idealities on DNNs, thereby ameliorating the performance (accuracy) of DNNs on crossbars. Note, these works do not pose a huge overhead of hardware-aware retraining of a pretrained DNN model before deployment on crossbars.

We begin by discussing two recent works that use analog crossbars and improve the adversarial robustness of the mapped DNNs. Note, adversarial robustness implies improving the performance of the hardware-mapped DNN model against adversarial samples without compromising the classification accuracy of clean images on hardware. In the first work, we introduce a technique called NEAT [26] that mitigates the impact of selector-induced non-linearities and resistive crossbar non-idealities for robust implementation of DNNs on 1T-1R crossbars. Second, we showcase another work, called DetectX [27], that uses hardware signatures present in analog crossbar architectures to perform energy-efficient adversarial detection. While NEAT is tailored for adversarial classification, DetectX is an adversarial detection method.

Finally, we delve into a specific case of the inference of DNNs having structured sparsity in their weights on analog crossbar arrays. Recently, crossbar-aware structured pruning algorithms [28–31] have received significant attention in developing increasingly sparse DNN models requiring fewer crossbars to be mapped, thereby introducing huge savings in terms of crossbar energy and area-efficiencies [32]. However, a holistic evaluation of the performance of such algorithms by considering the impact of resistive crossbar non-idealities was missing. In this chapter, we highlight a recent work [33] which shows that increased structured sparsity in DNNs negatively interferes with crossbar non-idealities that can degrade their classification accuracy (or robustness) during inference. This work also introduces two hardware-centric non-ideality mitigation strategies, namely crossbar-column rearrangement and Weight-Constrained-Training (WCT), to help improve the performance or robustness of the sparse DNNs on crossbars.

This chapter is organized as follows. Section 2 explains the background on adversarial attacks, memristive crossbars and non-idealities. In Sect. 3, we discuss the NEAT technique that introduces a non-ideality control technique which causes a rise in adversarial robustness. Section 4 introduces the DetectX technique that performs energy signature separation for adversarial detection. Section 5 explains the impact of structured sparsity in DNNs and their interaction with non-ideal crossbars. Section 6 gives an overview of related works and scopes out different crossbar-based studies with different objectives. Finally, we conclude in Sect. 7.

2 Background

2.1 Adversarial Attacks

Adversarial samples are created by generating a crafted noise and adding it to the clean data samples. In this chapter, we discuss two widely used methods to generate the noise for creating adversarial attacks.

1. *Fast Gradient Sign Method (FGSM)* [6] is a one-step gradient-based attack shown in Eq. (1). To generate the noise, first, the gradients of the DNN loss $\mathcal{L}(\theta, x, y_{true})$ with respect to the input x are calculated. Here, θ represents the parameters of the DNN and y_{true} represents the labels of the input data. Then, a $sign()$ operation converts the gradients into unit directional vectors. The unit vector is multiplied by a scalar perturbation value, ϵ, that determines the strength of the attack. Finally, the perturbation vector is added to the input x to create an adversarial data. Note that perturbations are added to x along the direction of the gradients to maximize DNN loss $\mathcal{L}$.

$$x_{adv} = x + \epsilon\ sign(\nabla_x(\mathcal{L}(\theta, x, y_{true}))) \tag{1}$$

2. *Projected Gradient Descent (PGD)*: The PGD attack, shown in Eq. (2) is an iterative attack over n steps. It is basically a multi-step variant of the FGSM attack. In each step i, perturbations of strength α are added to x_{adv}^{i-1}. Note that x_{adv}^{0} is created by adding random noise to the clean input x. Additionally, for each step, x_{adv}^{i} is projected on a *Norm ball* [8], of radius ϵ. In this chapter, the L_∞ *Norm ball* (of radius ϵ) projection is used for all the PGD attacks. In other words, we ensure that the maximum pixel difference between the clean and adversarial inputs is ϵ.

$$x_{adv} = \sum_{i=1}^{n} x_{adv}^{i-1} + \alpha\ sign(\nabla_x \mathcal{L}(\theta, x, y_{true})) \tag{2}$$

2.2 Memristive Crossbars and Their Non-idealities and Non-linearities

Memristive crossbar arrays have been used to implement MAC operations in an analog manner. Crossbars consist of 2D arrays of NVM devices, *Digital-to-Analog Converters* (DAC), *Analog-to-Digital Converters* (ADC) and a write circuit. The synaptic devices at the cross-points are programmed to a particular value of conductance (between G_{MIN} and G_{MAX}) during inference. The MAC operations are performed by converting the digital inputs to the DNN into analog voltages on the *Read Wordlines* (RWLs) using DACs, and sensing the output current flowing through the bit-lines (BLs) using the ADCs [23, 34–38]. In other words, the activations of the DNNs are mapped as analog voltages V_i input to each row and weights are programmed as synaptic device conductances (G_{ij}) at the cross-points as shown in Fig. 2a. For an ideal crossbar array, during inference, the voltages interact with the device conductances and produce a current (governed by Ohm's Law). Consequently, by Kirchhoff's current law, the net output current sensed at each column j is the sum of currents through each device, i.e. $I_{j(ideal)} = \Sigma_i G_{ij} * V_i$. We term the matrix G_{ideal} as the collection of all G_{ij}'s for a crossbar instance. However, in reality, the analog nature of the computation leads to various hardware noise or non-idealities, such as, circuit-level resistive non-idealities and device-level variations [23, 34, 36, 37, 39–43].

Non-idealities: Fig. 2b describes the equivalent circuit for a crossbar accounting for various circuit-level and device-level non-idealities, *viz.* $Rdriver$, $Rwire\_row$, $Rwire\_col$ and $Rsense$ (interconnect parasitics), modelled as parasitic resistances and variations in the synapses owing to the stochasticity of the memristive devices. This results in a $G_{non\text{-}ideal}$ matrix, with each element G'_{ij} incorporating the effect due to the non-idealities, obtained using circuit laws (Kirchhoff's laws and Ohm's law) and linear algebraic operations [23, 24, 33, 39, 44]. Consequently, the net output current sensed at each column j becomes $I_{j(non\text{-}ideal)} = \Sigma_i G'_{ij} * V_i$, which deviates

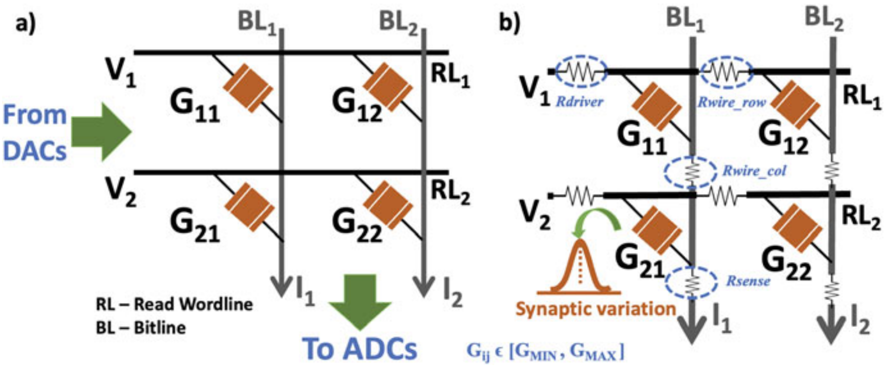


Fig. 2 (**a**) A 2 × 2 crossbar array with input voltages V_i, synaptic conductances G_{ij} and output currents $I_j = \sum_i G_{ij} * V_i$. (**b**) A 2 × 2 crossbar array with the resistive and the synaptic device-level non-idealities. These non-idealities lead to imprecise dot-product currents and that manifests as accuracy degradation when DNNs are evaluated on crossbars

from its ideal value. This manifests as accuracy degradation for DNNs mapped onto crossbars. The relative deviation of $I_{non\text{-}ideal}$ from its ideal value is measured using *non-ideality factor* (NF) [34] as:

$$NF = (I_{ideal} - I_{non\text{-}ideal})/I_{ideal}. \tag{3}$$

Thus, NF is a direct measure of crossbar non-idealities, i.e. increased non-idealities induce a greater value of NF, affecting the accuracy and hence, the robustness of the DNNs mapped onto them. All the analyses in Sects. 3 and 5 involving non-idealities are performed on memristive crossbars with an ON/OFF ratio of 10 (i.e. $R_{MIN} = 30\,k\Omega$ and $R_{MAX} = 300\,k\Omega$), having the resistive non-idealities as follows: $Rdriver = 1\,k\Omega$, $Rwire\_row = 5\,\Omega$, $Rwire\_col = 10\,\Omega$ and $Rsense = 1\,k\Omega$. The non-ideality in the memristive devices in the form of device-level process variation has been modelled as a Gaussian variation in the conductances (G_M) of the NVM devices with $\sigma/\mu = 10\%$ [45].

Recently, 1T-1R NVM crossbars have received significant attention since the pass-transistor in series with the NVM device at the synapses can help mitigate sneak paths (prevalent in 1R crossbar arrays) and the incorrect programming of the NVM device induced by noise [46, 47]. Figure 3 illustrates an $M \times N$ crossbar having 1T-1R synapses at the cross-points wherein, the access transistors are driven by a gate-voltage (V_g) fed through the select-lines (SLs). A low V_g operation is favorable for implementing a DNN on crossbars in a resource-constrained scenario as it has been shown in prior works [26] that the total power dissipated by a 1T-1R crossbar array diminishes with reduction of the transistor gate-voltage (V_g). However, it is imperative to understand the other repercussions of low V_g operation in 1T-1R crossbars that impact the performance and hence, robustness of the mapped DNN models.

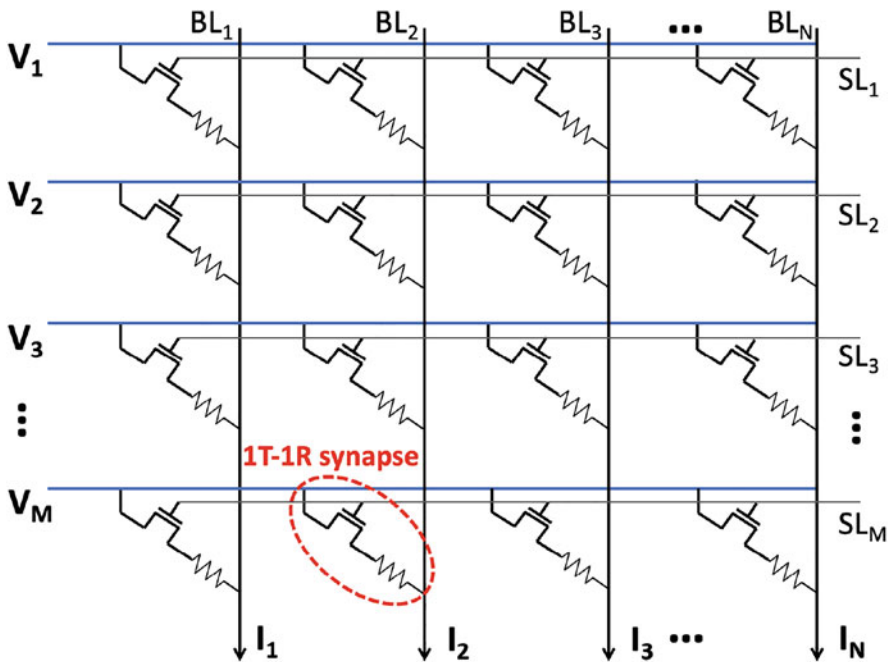


Fig. 3 Illustration of an M × N 1T-1R Crossbar. A Transistor (T) in series with an NVM device (R) is present at every synapse. Select-lines (SLs) are used to turn on transistors for selected rows, while the dot-product currents are sensed through the bit-lines (BLs)

Non-linearities: In addition to the above-mentioned non-idealities, 1T-1R crossbars are susceptible to various non-linearities that affect the effective conductance of each synapse (especially, at lower V_g) and hence, the output current across each column in a crossbar array. This would manifest as accuracy degradation for the DNNs mapped onto such crossbars. In [26], to understand the effects of the non-linearities alone on introducing the access transistor (or selector) in the synapse, extensive SPICE simulations were performed using the 1T-1R synaptic configuration with different input voltages, conductances and V_g ranges excluding the circuit-level and device-level non-idealities. For all the analyses involving 1T-1R synapses, the selector devices were based on 45 nm CMOS technology model and the memristive device had $R_{ON} = 30\,k\Omega$ and $R_{OFF} = 300\,k\Omega$.

For a crossbar in the 1R configuration, the weights W of the DNN are directly mapped to a memristor conductance state ($G_M = 1/R_M$) in a linear fashion. On the other hand, in the 1T-1R configuration, W is mapped to the effective conductance $G_{eff} = 1/(R_M + R_t)$, where R_t is the equivalent resistance due to the transistor. The non-linearities in the 1T-1R crossbars arise due to the dependence of R_t on V_{in}. Note, V_{in} is proportional to the neuronal activation values of the DNN which varies with the input. Hence, these are data-dependent non-linearities. It has been shown in [26] that the effective conductance G_{eff} is a function of NVM conductance G_M, input voltage V_{in}, and gate-voltage V_g, which can be formulated as:

$$G_{eff} = f_1(G_M, V_{in}, V_g). \tag{4}$$

3 Non-linearity Aware Training (NEAT): Mitigating the Impact of Crossbar Non-idealities and Non-linearities for Robust DNN Implementations

In this section, the NEAT technique is introduced that provides a new perspective on the energy-efficient and robust implementation of DNNs on 1T-1R crossbars [26]. It begins with the identification of a range of memristive conductances over a range of input voltages via SPICE simulations such that Eq. (4) can be approximated as:

$$G_{eff} = f_2(G_M, V_g). \tag{5}$$

This eliminates the input-data dependency of the effective 1T-1R synaptic conductance (G_{eff}) for a given value of V_g of transistor operation. In other words, for a given value of V_g, there exists an upper bound cut-off value ($G_{eff\,cutoff}$) for which the 1T-1R synapse exhibits linear characteristics, and $G_M \approx k * G_{eff}$, where k is a scalar. The corresponding NVM device state at $G_{eff\,cutoff}$ is termed as $G_{M\,cutoff}$. Figure 4 shows the $G_{M\,cutoff}$ vs. V_g plot for supply voltage $V_{supply} = 0.25V,\ 0.5V$ and V_{in} in the range $0 \leq V_{in} \leq V_{supply}$. It can be seen that as V_g is lowered (for resource-constrained scenarios), the overall range of memristive conductance states G_M for data-independent and linear synaptic characteristics decreases owing to low values of $G_{M\,cutoff}$.

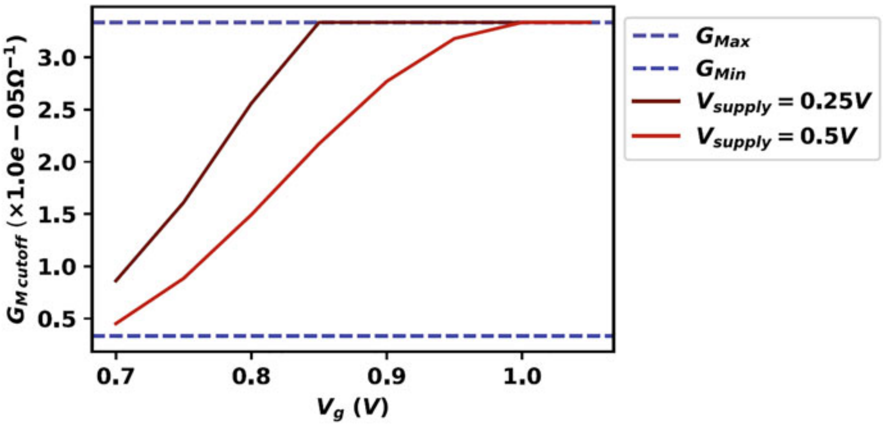


Fig. 4 The variation in $G_{M\,cutoff}$ with respect to selector gate-voltage V_g

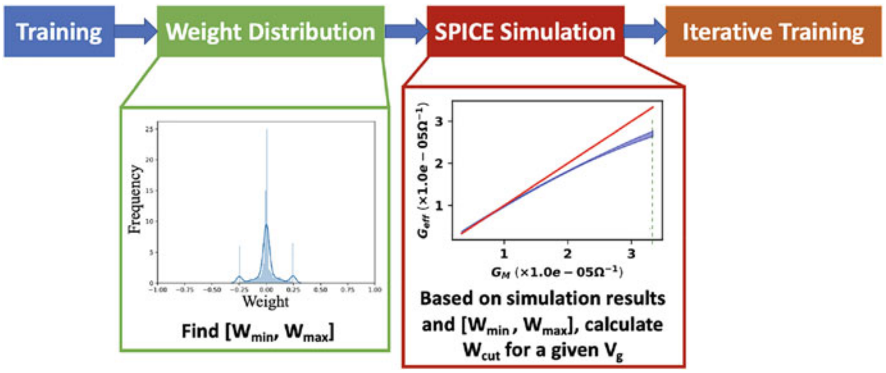


Fig. 5 Overall flow of NEAT

After identifying $G_{eff\,cutoff}$ for a given V_g, the corresponding value for W_{cut} is obtained for the software DNN which is to be mapped onto the 1T-1R crossbars. Then, all the weights (W) of the pretrained DNN are restricted in the interval $[-W_{cut}, W_{cut}]$ as shown in Eq. (6):

$$W_{map} = \begin{cases} W & |W| \leq W_{cut} \\ W_{cut} & W > W_{cut} \\ -W_{cut} & W < -W_{cut}. \end{cases} \tag{6}$$

From Eq. (6), we observe that for the linear regime ($|W| \leq W_{cut}$, which corresponds to $G_{eff} \approx k * G_M$), the software weight parameters can be mapped linearly onto the crossbars. While, for the non-linear regime ($|W| > W_{cut}$ that corresponds to deviation of G_{eff} from G_M), W is clipped at W_{cut}. The objective of NEAT is to restrict the weight parameters to be within the linear regime for the given gate-voltage V_g of the transistor, thereby curbing loss in computational accuracy post-mapping DNNs onto 1T-1R crossbars. Figure 5 illustrates the overall flow of the NEAT process.

Iterative Training: In NEAT, after setting the optimal V_g and W_{cut} values, the weights of the DNN get transformed. If we use lower values of V_g which do not cover all weight ranges, the weight distribution gets altered, resulting in accuracy degradation. To address this issue, iterative training is proposed which consists of two steps. Step 1 is essentially restricting the weights of the DNN (W) in the suitable cut-off regime as per Eq. (6). Step 2 involves retraining the networks iteratively for a couple of epochs to recover any accuracy loss incurred from Step 1. These two steps are repeated so that greater number of weights in the network can be located in the linear regime when mapped onto crossbars.

In Fig. 6, V_g is varied from $0.75\,V$ to 1.0 and the classification accuracy of various DNN architectures using CIFAR10 and CIFAR100 datasets is reported. The results show that low V_g induces low W_{cut} and in turn decreases performance

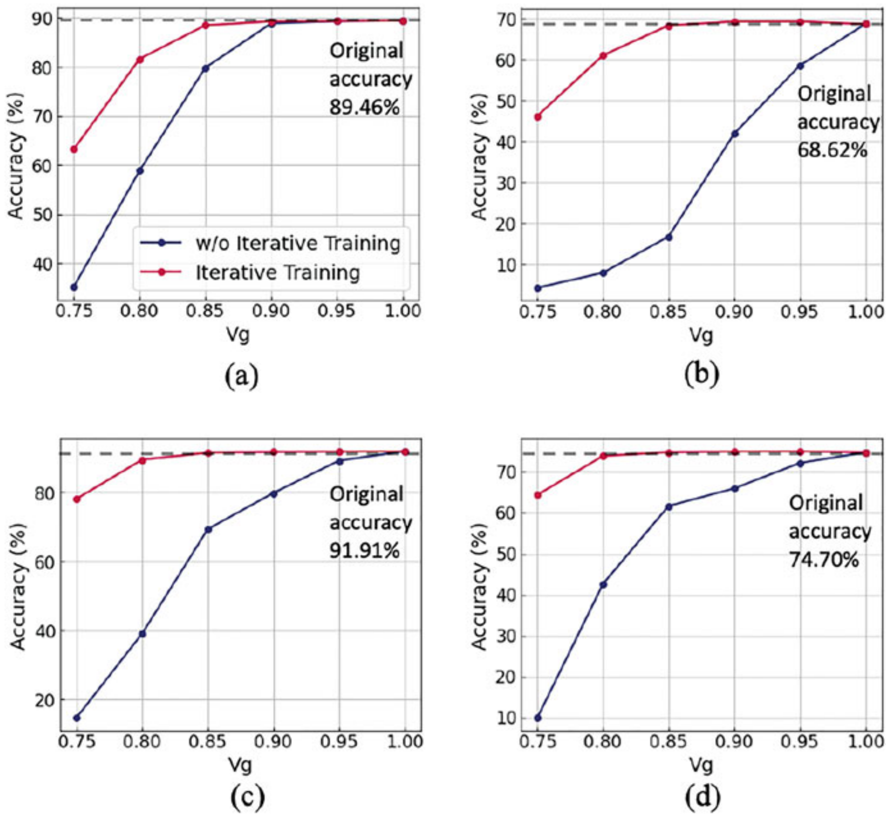


Fig. 6 Classification accuracy of various NEAT-based DNN models with V_g varied from 0.75 V to 1.0 V. (**a**) VGG11/CIFAR10. (**b**) VGG11/CIFAR100. (**c**) ResNet18/CIFAR10. (**d**) ResNet18/CIFAR100

when DNN weights are restricted to W_{cut} regime. However, using iterative training recovers the performance degradation. Especially, for a ResNet18 architecture, using iterative training shows improvement over 50% in terms of accuracy at $V_g = 0.75\,V$. Moreover, with iterative training, VGG11 and ResNet18 networks almost maintain their classification accuracy in the range of $V_g = [0.85, 1.0]$ and $V_g = [0.8, 1.0]$, respectively. In this manner, NEAT helps in the hardware-aware robust mapping of DNN architectures on 1T-1R crossbar with minimal training overheads.

In addition to maintaining DNN performance in presence of 1T-1R non-linearities, NEAT ensures energy-efficient inference with DNNs, specifically in the low V_g scenario. When NEAT technique is applied at low V_g values, we have lower absolute values of $G_{eff\,cutoff}$ and hence, W_{cut}. This implies that for crossbars operating at lower V_g values, we would find greater proportion of low conductance synapses post mapping of DNN weights onto the 1T-1R crossbars. This helps

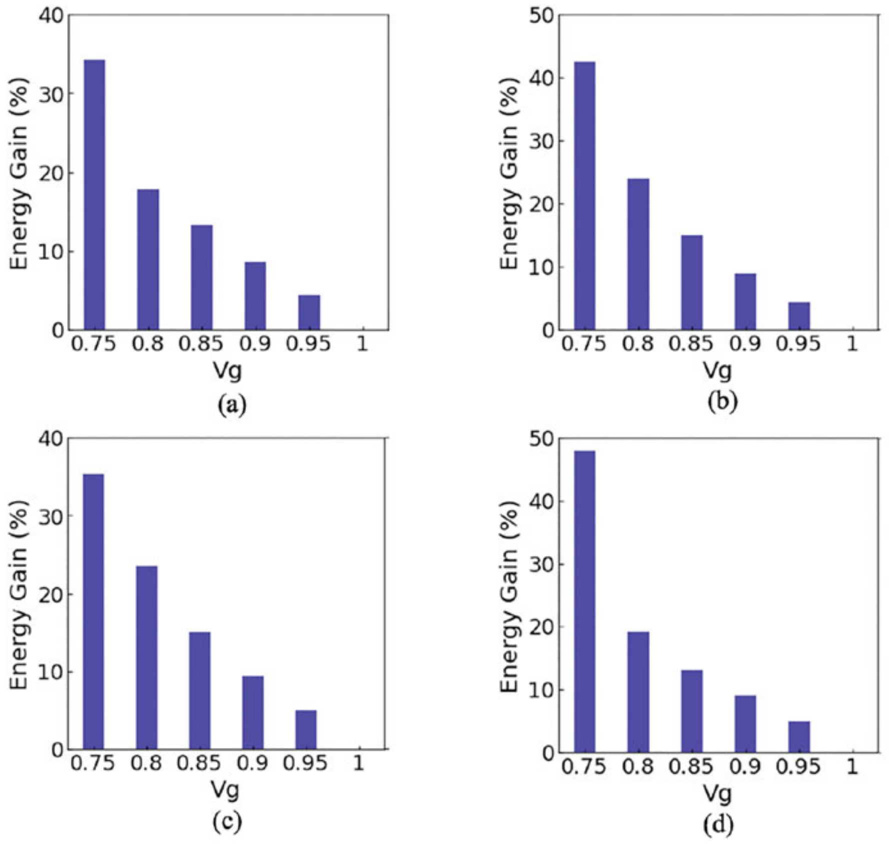


Fig. 7 Normalized energy gain for various NEAT-based DNN models with V_g varied from 0.75 V to 1.0 V. (**a**) VGG11/CIFAR10. (**b**) VGG11/CIFAR100. (**c**) ResNet18/CIFAR10. (**d**) ResNet18/CIFAR100

minimise the power dissipated in the crossbar arrays by reducing crossbar-column currents. In Fig. 7, we present the energy-efficiency of various DNN configurations with NEAT. The energy computed for $V_g = 1.0\,V$ is taken as *baseline* against which energy gains (%) for other values of V_g are shown. NEAT achieves high energy gain by simply reducing V_g. Especially, we can achieve ~23% energy gain at $V_g = 0.8\,V$ on ResNet18 architecture with CIFAR10 while suffering minimal accuracy loss (~1.5% in Fig. 6). However, selecting a very low value for V_g such as $V_g = 0.75\,V$ induces huge performance degradation.

Having mapped DNNs via NEAT in an energy-efficient manner onto 1T-1R crossbars and mitigating the impact of synaptic non-linearities, we now study the impact of crossbar non-idealities (enlisted in Sect. 2.2) on the robustness of NEAT-DNNs. Note, we would use the term "NEAT-DNN" to denote a DNN trained using NEAT and mapped onto 1T-1R crossbars, while the term 'Normal-DNN' would refer to standard DNNs mapped directly onto 1R crossbars with non-idealities (the *baseline*). Henceforth, all experiments in Sect. 3 involving crossbar arrays would include the crossbar non-idealities.

It has been shown that the value of NF in crossbars decreases with increase in the effective resistance of a crossbar array, which minimizes the effect of interconnect resistive non-idealities [26, 33, 34, 44]. By increasing the proportion of lower conductance synapses in a crossbar array, one can reduce the impact of crossbar non-idealities and hence, the non-ideality factor. When NEAT technique is applied at low V_g values, we have a greater proportion of low conductance synapses post mapping of DNN weights onto the 1T-1R crossbars. Hence, NF is expected to be lower in the case of low V_g operation of crossbars. Furthermore, the value of NF for DNNs mapped onto 1T-1R crossbars via NEAT would be lesser than the value of NF for standard DNNs mapped onto 1R crossbars (the *baseline*). Since, NEAT boosts the feasibility of low conductance synapses and reduces the impact of crossbar non-idealities, NEAT-DNNs are more robust both in terms of clean and adversarial accuracies than the *baseline* Normal-DNNs.

Modes of adversarial attack: For unleashing adversarial attacks (FGSM/PGD) on the crossbar-mapped models of the DNNs, consider two modes:

1. **Software-inputs-on-hardware (SH) mode** where, the adversarial perturbations for each attack are created using the loss function of the software DNN model normally trained without applying the NEAT technique, and then added to the clean input that yields the adversarial input. The generated adversaries are then fed to the crossbar-mapped DNN. This a case of Black-Box adversarial attack on hardware.
2. **Hardware-inputs-on-hardware (HH) mode** where, the adversarial inputs are generated for each attack using the loss from the crossbar-based hardware models. It is evident that HH perturbations will incorporate the effect of intrinsic hardware non-idealities and thus will cast stronger attacks than SH. This is a case of White-Box adversarial attack on hardware.

Results for robustness in terms of clean and adversarial accuracies: Here, we evaluate robustness of the DNNs on non-ideal crossbars graphically as shown in Fig. 8 by plotting *'robustness maps'* as has been proposed in [26, 44]. Note, the NEAT-DNNs are shown for $V_g = 0.8\,V$. This approach to assess robustness of a network has been shown to be comprehensive and accurate since, it takes into account the cumulative impact of both clean accuracy and adversarial accuracy (which is a strong function of the clean accuracy). For a specific mode of attack (SH or HH) and a given crossbar size, we plot $\Delta\ Clean\ Accuracy$, the difference between clean accuracy of the crossbar-mapped DNN in question and the corresponding clean accuracy of its software model, on the x-axis. $\Delta\ Adversarial\ Accuracy$ (for a particular ϵ value) which is the difference between the adversarial accuracy of the mapped network in question and the corresponding adversarial accuracy of the software model is plotted on the y-axis. The value of $\Delta\ Clean\ Accuracy$ is always negative since DNNs when mapped on hardware suffer accuracy loss owing to non-idealities. The region bounded by the line $y = -x$

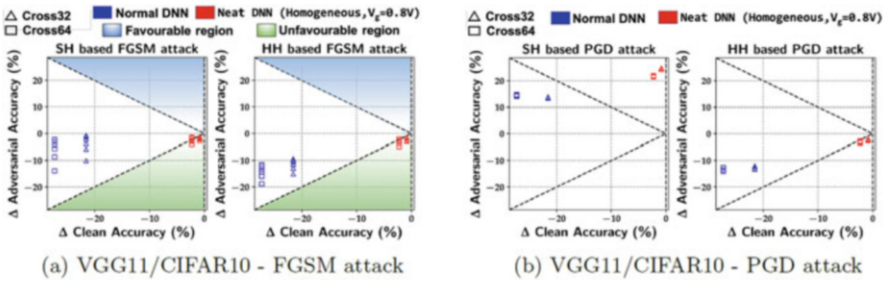


Fig. 8 (**a**), (**b**) Robustness maps for VGG11 DNN using CIFAR10 dataset for SH and HH modes of FGSM and PGD attacks, respectively

and the y-axis denotes the *favorable region* and the closer a point is towards this region, the better is the robustness of the network in question. Likewise, the region bounded by the line $y = x$ and the y-axis is the *unfavorable region*, where the mapped network is highly vulnerable to adversarial attacks. The favorable and unfavorable regions have been demarcated Fig. 8a.

Figure 8 shows the robustness maps for DNNs based on VGG11 network with CIFAR10 dataset for both SH and HH modes of attack. Figure 8a pertains to FGSM attack with ϵ varying from 0.05 to 0.3 with step size of 0.05. We find that NEAT-DNNs have significantly greater clean accuracy ($\sim$13% and $\sim$17% higher for 32×32 and 64×64 crossbars, respectively) as well as better adversarial accuracies on hardware for both modes of attack. The points corresponding to NEAT-DNNs are situated closer to the favorable region than the corresponding points for Normal-DNNs. This is a consequence of the reduction in non-ideality factor in case of iterative training with NEAT algorithm. Note, the points for 64×64 crossbars are situated farther from the favorable region than the corresponding points for 32×32 crossbars. This gap is owing to greater non-idealities that exist in case of a larger 64×64 crossbar than a 32×32 crossbar. However, this gap significantly decreases for NEAT-DNNs indicating that NEAT greatly reduces the impact of the crossbar non-idealities on the inference accuracy of the mapped DNNs. In other words, NEAT-DNNs do not suffer significant accuracy losses on larger crossbars. We further observe that the points for a NEAT-DNN, given a crossbar size, are more closely packed than the corresponding points for Normal-DNN. This implies that even on increasing the attack strength (ϵ), lesser adversarial loss is observed for DNN models on crossbars trained with NEAT algorithm.

Figure 8b also presents similar results but for a strong PGD attack with ϵ varying from 2/255 to 32/255 with step size of 2/255. In this case, the robustness is very high for SH mode of attack as compared to HH mode of attack, with points corresponding to NEAT-DNNs situated inside the favorable region for the SH mode. Similar to the case of FGSM attack, NEAT-DNNs outperform Normal-DNNs in terms of robustness for both modes of attack. Here, points for different ϵ values, given a style of mapping and crossbar size, are more closely packed than the corresponding points of FGSM attack. This implies that hardware non-idealities interfere more with PGD attacks than FGSM attacks resulting in lesser accuracy loss.

From the above discussion, we find that NEAT-based DNNs are more immune to the impact of non-idealities and lead to robust implementations on non-ideal 1T-1R crossbars in addition to higher crossbar energy-efficiencies.

4 DetectX: Improving the Robustness of DNNs Using Hardware Signatures in Memristive Crossbar Arrays

In this section, we discuss how hardware signatures in memristive crossbar architectures can be used to detect adversarial attacks in an energy-efficient manner [27].

For detecting adversaries, we use a function called the Sum of Currents (SoI). It is defined as the absolute value summation of all the feature outputs of a particular layer as shown in Eq. (7).

$$SoI_l = \sum_{j=1}^{m} |Z_j|_l \tag{7}$$

Here, Z_j is the result of the weighted summation outputs of a particular layer l. This is proportional to the summation of the column current magnitudes in a memristive crossbar array. Interestingly, as shown in Fig. 9a, we find that the clean and adversarial SoI distributions of the first layer have an inherent separation between them. However, due to a significant overlap between the two distributions, the adversarial detection is low. To this end, we use a dual-phase training methodology to increase the distance between the SoI distributions and improve the adversarial detection.

In the first phase of training, we train the first layer of the DNN to increase the separation between the clean and adversarial SoIs. For this, we use a loss function

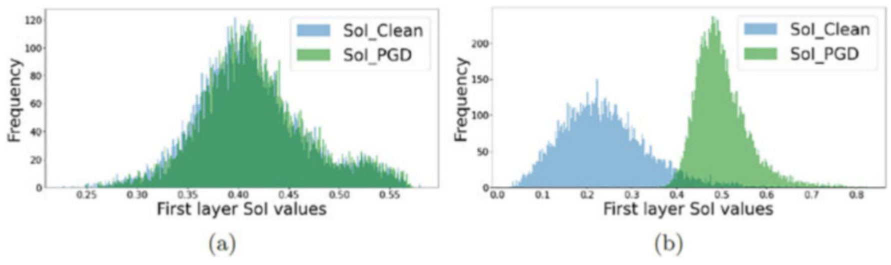


Fig. 9 (**a**) The clean and adversarial SoI distributions at the first layer have an inherent separation which motivates the use of SoI like hardware signature for adversarial detection. (**b**) After Phase1 training, the SoI distributions are separated. For both the figures, SoI PGD corresponds to first layer SoI values for PGD with $\epsilon = 16/255$

shown in Eq. (8). Here, we scale down the cross-entropy loss $\mathcal{L}_{CE}$ by a small value (of the order 10^{-3}). The loss function minimizes the distance between the desired SoI values (λ_c and λ_a) and the calculated mean of the SoI distributions (SoI_c and SoI_a).

The Phase1 training effectively increases the distance between the clean and adversarial SoI distributions as seen in Fig. 9b. At this stage, strong adversarial attacks are easily detected as a result of large SoI separation. However, weak attacks are not sufficiently detected as they have small SoI separation with the clean samples. Note, here strong attacks refer to adversarial attacks with high L_∞ distance (large ϵ value) and vice versa. Further, the DNN has very low accuracy on clean inputs as the cross-entropy loss was significantly scaled down during Phase1 training. To improve the DNN's accuracy on clean inputs and robustness against weak adversarial attacks, we employ Phase2 adversarial training.

$$\mathcal{L} = \beta\mathcal{L}_{CE} + y\mathcal{L}_{MSE}(SoI_a, \lambda_a) + (1-y)\mathcal{L}_{MSE}(SoI_c, \lambda_c) \tag{8}$$

In the Phase2 training, we freeze the first layer of the DNN and perform adversarial training [8, 48] with weak adversarial attacks. Freezing the first layer weights preserves the SoI separation at the first layer obtained after Phase1 training. Finally, after the dual-phase training, a high clean accuracy is obtained. Additionally, weak adversarial attacks are suitably classified while strong attacks are detected.

After the Phase2 training, we create a SoI-Probability Look-Up Table (LUT) that classifies a given SoI value as a clean or an adversarial sample. As seen in Fig. 10, we randomly sample a set of clean images from the training set and create their adversarial counterparts using PGD $\epsilon = 8/255$ attack. Then, we compute the clean and adversarial SoI distributions ($\mathcal{D}_c$ and $\mathcal{D}_a$). Using, $\mathcal{D}_c$ and $\mathcal{D}_a$, we compute the P(clean) values using Eq. (9). Here, n_c and n_a are the number of clean and adversarial samples, respectively, at a particular SoI value. A high *P(clean)* value

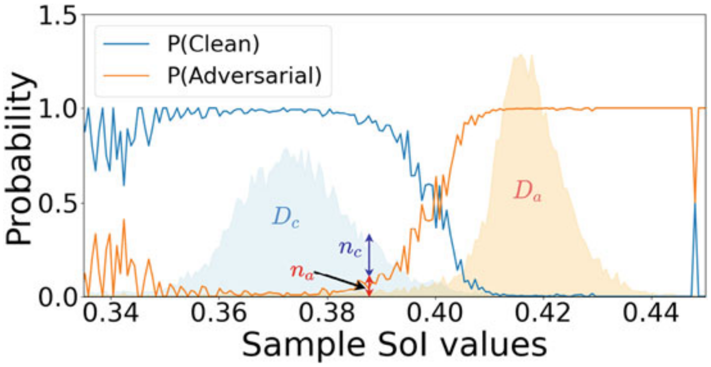


Fig. 10 The SoI-Probability LUT contains sample SoI values and their corresponding P(clean) values. It classifies a given SoI sample as clean or adversarial

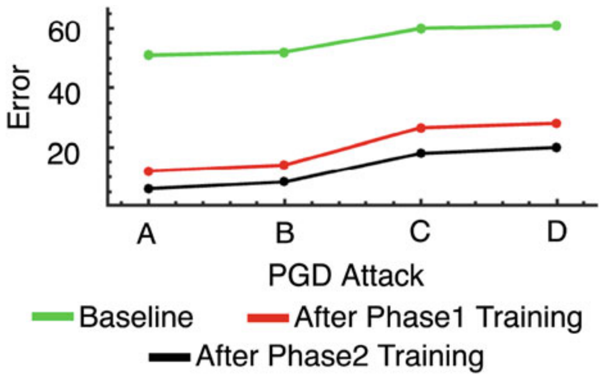


Fig. 11 After Phase1 training, some weak adversarial attacks might go undetected. Phase2 adversarial training helps classify the weak adversarial samples that further brings down the error

signifies that a given SoI value corresponds to a clean sample and vice versa. The SoI-Probability LUT contains sample SoI values and their corresponding *P(clean)* values.

$$P(Clean) = \frac{n_c}{n_c + n_a} \tag{9}$$

In Fig. 11, we show the efficacy of the Phase2 training in defending against weak PGD attacks. For this we plot the error values of a baseline DNN (without DetectX), a DNN with the first layer subjected to Phase1 training followed by Phase2 training. Here, error is defined as the amount of adversarial attacks that are undetected and are misclassified by the DNN. Clearly, most of the weak attacks are suitably detected after Phase1 training leading to a large drop in the error value. However, Phase 2 adversarial training helps classify the undetected weak attacks correctly leading to a further drop in error.

We implement the dual-phase trained DNN on an analog crossbar-based end-to-end DNN evaluation platform called Neurosim [49]. Neurosim [49] is a Python-based platform that performs a holistic energy-latency-accuracy evaluation of analog crossbar-based DNN accelerators. The Neurosim platform supports both SRAM and memristive computing devices (ReRAM and FeFET). For adversarial detection, we design a fully digital DetectX module (shown in Fig. 12) on 32 nm CMOS that contains digital circuits to compute the SoI value and the SoI-Probability LUT that is used to classify a given SoI value as clean or adversarial. For hardware evaluation, a dual-phase trained VGG16 model (trained on CIFAR100) is implemented on a 128×128 memristive crossbar with device on-off ratio of 10 and $R_{on} = 10\,\mathrm{k}\Omega$. The DetectX module is appended at the end of the first layer crossbar. The energy evaluation of the DetectX module is performed using SPICE simulations. Based on the 128×128 crossbar Neurosim implementation, we find that the DetectX module only adds 2.6 nJ to the hardware cost for adversarial

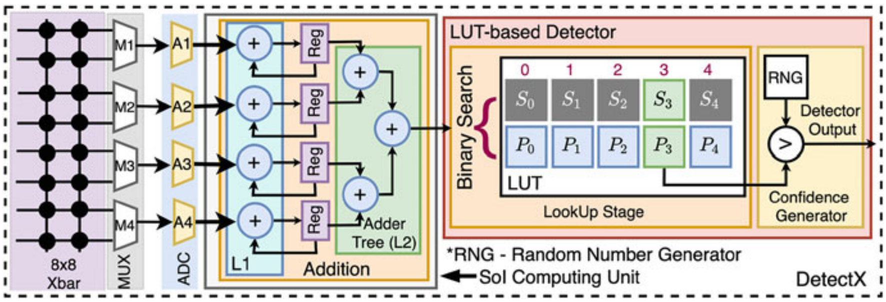


Fig. 12 The DetectX module is implemented on a fully digital 32 nm CMOS technology. It can directly interface with an analog crossbar (8 × 8 crossbar shown for illustration). The module contains circuits for computing the SoI signature and classifying the SoI value as clean or adversarial

CIFAR10 VGG8 Phase1- PGD (ϵ = 32/255), Phase2 – PGD (ϵ = 4/255)				
	FGSM (ϵ = 16/255)	PGD (ϵ = 8/255)	PGD (ϵ = 16/255)	PGD (ϵ = 32/255)
ROC-AUC Score	0.97	0.895	1	1
Accuracy (D/B)	80.1 / 87			
Error (D/B)	1.7 / 35.3	9.1 / 53.2	0 / 66.7	0 / 67.18
CIFAR100 VGG16 Phase1- PGD (ϵ = 32/255), Phase2 – PGD (ϵ = 4/255)				
ROC-AUC Score	0.98	0.99	1	1
Accuracy (D/B)	50.1 / 60.2			
Error (D/B)	5.1 / 59.4	0 / 96.6	0 / 97.7	0 / 97.7
TinyImagenet ResNet18 Phase1- PGD (ϵ = 32/255), Phase2 – PGD (ϵ = 4/255)				
ROC-AUC Score	0.84	0.86	0.98	1
Accuracy (D/B)	42.2 / 53.3			
Error (D/B)	13.2 / 56.5	12.36 / 90.7	10.1 / 91.2	0.1 / 93

Fig. 13 ROC-AUC scores, Error and Accuracy values for the DNN + DetectX system with different image datasets and adversarial attacks. We mention the adversarial attacks used for Phase1 and Phase2 training corresponding to each dataset. *D* and *B* denote the DNN + DetectX system and baseline model, respectively. The baseline model is a DNN trained on clean inputs using standard stochastic gradient descent and does not contain the DetectX module

detection. Compared to prior adversarial detection works that use large neural network-based detector modules [16–18], DetectX consumes about 25x less energy for adversarial detection.

DetectX significantly improves the adversarial robustness of the DNN. In Fig. 13, we show the ROC-AUC score of the DetectX module under different FGSM and PGD attacks across CIFAR10, CIFAR100 and TinyImagenet datasets. A high ROC-AUC score greater than 0.5 denotes reliable adversarial detection. Due to the high ROC-AUC score, the error of the DNN + DetectX system is significantly lower compared to the baseline DNN without the DetectX module. Further, due to the introduction of the DetectX module, the accuracy on clean inputs slightly drops. However, the drop is marginally low.

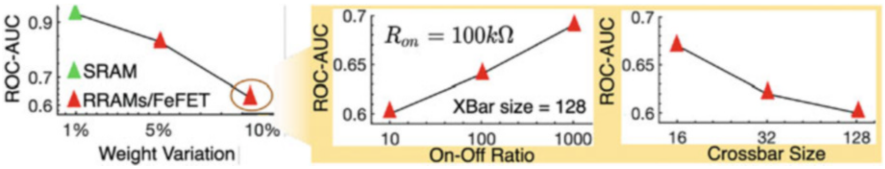
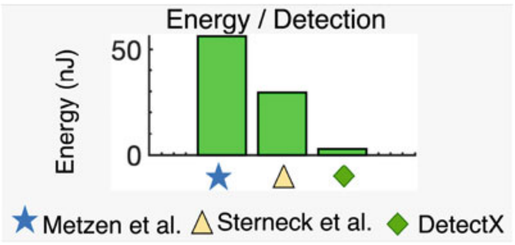


Fig. 14 With increasing device-device variations in a memristive crossbar (left), the adversarial detection performance decreases slightly. Further, with increasing device on-off ratio (middle), and decreasing crossbar sizes (right), the detection performance increases. Non-idealities negatively impact the detection performance of DetectX

Fig. 15 Energy required per detection operation for different works [16, 18]. DetectX consumes more than 25x less energy for detection compared to prior works



As DetectX is integrated in a crossbar platform like Neurosim, it is important to understand the effects of crossbar non-idealities on the detection performance of DetectX. Figure 14(left) shows that the detection performance decreases with increasing device-device variations in the memristive crossbars [50]. The device variations introduce variations in the SoI value computation which ultimately negatively affect the detection performance. However, even at large weight variations, the ROC-AUC score is still greater than 0.5 which suggests reliable detection. Next, we show the effects of different memristive device on-off ratios (Fig. 14(middle)) and crossbar sizes (Fig. 14(right)) on DetectX's detection performance. These results are shown for the memristive device with 10% weight variations (shown with a circle). Evidently, the detection performance increases with higher on-off ratios and lower crossbar sizes. This is because with higher on-off ratios and lower crossbar sizes, the non-ideal effects in crossbars decrease.

We further show how the DetectX method consumes significantly low energy for adversarial detection compared to prior detection works [16–18]. Prior works use large neural networks to perform adversarial detection. For a fair comparison, these neural network-based detectors are implemented on the Neurosim platform [49] and the energies are evaluated. As seen in Fig. 15 DetectX consumes about 50x less energy compared to Metzen et al. [16] and 25x less energy compared to Sterneck et al. [18]. Note, here the energy values represent the energy required for a single detection operation.

5 Unleashing Robustness to Structure-Pruned DNNs Implemented on Crossbars with Non-idealities

In the recent years, several crossbar-aware pruning techniques have been devised that yield sparse DNN models. Owing to their high sparsity, these models require significantly lower number of crossbars to be mapped, thereby introducing hardware resource-efficiency not only in terms of crossbars but also peripheral circuits interfacing the crossbars. Pruning algorithms such as, [28–31], produce structured sparsity in DNNs that fit into crossbars as dense weight matrices [32]. These structured pruning algorithms claim to preserve the accuracy of the pruned DNNs, after implementation on crossbars, with minimal or no noticeable loss, while bringing in high energy- and area-efficiencies. However, none of these works has included the impact of the inexorable non-idealities (see Sect. 2.2) during inference on crossbars. For a realistic hardware evaluation of the performance of increased structured sparsity in DNNs mapped on crossbars, the inclusion of hardware non-idealities is critical. In this section, we introduce a recent work [33] that draws the focus of the research community towards a non-ideality aware evaluation of various existing structured pruning algorithms and shows how increased sparsity can degrade the robustness of DNNs on non-ideal crossbars. It also introduces two hardware-centric non-ideality mitigation strategies, namely crossbar-column rearrangement and *Weight-Constrained-Training* (WCT), to help improve the performance or robustness of the sparse DNNs on crossbars with little or no training overheads.

Crossbar-aware structured pruning of DNNs: There have been numerous works on structured pruning of DNNs, such as *channel/filter pruning* or C/F pruning (see Fig. 16(top)) wherein the unimportant filters and channels in a DNN (corresponding to rows and columns in the weight matrix of the DNN) are pruned to obtain a sparse 2D weight matrix [28, 29]. These pruned models result in significant hardware savings in terms of reduced number of crossbars for mapping, thereby bringing in energy- and area-efficiency for DNN implementation. Likewise, other crossbar-aware pruning strategies include *Crossbar-Column Sparsity* (XCS) [30] or *Crossbar-Row Sparsity* (XRS) [31] (XCS shown in Fig. 16(bottom)) that exploit fine-grained sparsity by, respectively, pruning columns or rows of weights within a crossbar [32]. Additionally, these works have claimed to preserve the inference accuracy of the structure-pruned networks on crossbars with minimal or no discernible performance loss with respect to the unpruned ones. However, none of the previous works has accounted for the non-idealities inherent in crossbar arrays which raises concerns about the claimed performance of the highly pruned models in the real scenario.

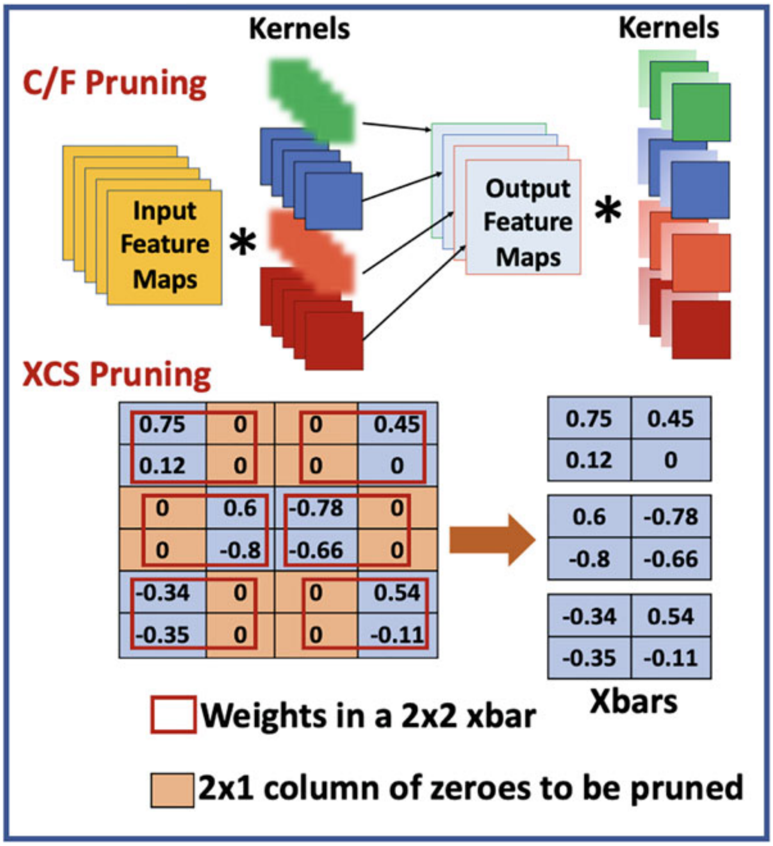
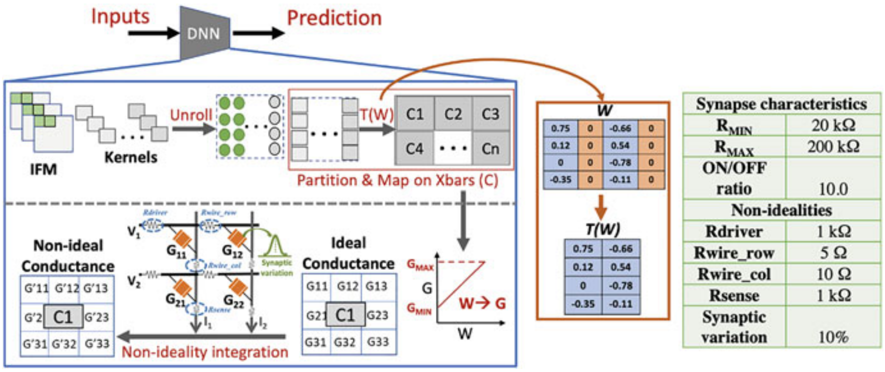


Fig. 16 **Top**: A representation of channel/filter pruning (C/F pruning). The blurred channels/filters correspond to DNN weights pruned in a structured manner. **Bottom**: A representation of XCS pruning (shown for a 4 × 4 weight matrix mapped onto 2 × 2 crossbars) that generates fine-grained sparsity along crossbar columns

5.1 Hardware Evaluation Framework for Non-ideality Integration During Inference

To map pretrained DNNs onto non-ideal memristive crossbars and investigate the cumulative impact of the circuit and device-level non-idealities on their performance during inference, a simulation framework in PyTorch is used by Bhattacharjee et al. [33] as shown in Fig. 17). In the platform, a Python wrapper is built that unrolls each and every convolution operation in the software DNN into MAC operations. This yields 2D weight matrices for each DNN layer which are to be partitioned into numerous crossbar instances. Before partitioning, based on the structured pruning approach, the following transformations T on the sparse weight matrices W are applied:



Synapse characteristics	
R_{MIN}	20 kΩ
R_{MAX}	200 kΩ
ON/OFF ratio	10.0
Non-idealities	
Rdriver	1 kΩ
Rwire_row	5 Ω
Rwire_col	10 Ω
Rsense	1 kΩ
Synaptic variation	10%

Fig. 17 Python-based hardware evaluation framework for non-ideality aware DNN inference

1. $T(W)$ **for C/F pruning:** Here, for a given 2D weight matrix of a DNN layer, all the columns bearing zero values are eliminated. Further, we also eliminate rows of the weight matrix of the next DNN layer that interact with the output feature maps corresponding to the columns of zero values in the previous layer.
2. $T(W)$ **for XCS (or XRS):** Here, within a given 2D weight matrix of a DNN layer, there are chunks of successive zero weight vectors of the size of crossbar-column (or crossbar-row) (see Fig. 16-Bottom) which are eliminated.

Note, for standard unpruned DNNs, the $T(W)$ is not required. The resulting transformed weight matrices are then partitioned into multiple crossbar instances. The subsequent stage of the platform converts the weights in the crossbars to suitable conductances G (between G_{MIN} and G_{MAX}). Thereafter, the circuit-level non-idealities (interconnect parasitics) and synaptic device variations are integrated with the conductances. The various synapse parameters (e.g. R_{MIN}, R_{MAX}, device ON/OFF ratio) and values of the non-idealities used for the subsequent experiments are listed in the table shown in Fig. 17.

5.2 *Are Structure-Pruned DNNs Also Robust on Hardware?*

In [33], VGG11 and VGG16 DNNs area trained with structured sparsity (via C/F pruning, XCS or XRS) using benchmark datasets such as, CIFAR10 and CIFAR100. For the experiments with CIFAR10 dataset, the sparsity is set as $s = 0.8$, while with CIFAR100 dataset, the sparsity is $s = 0.6$. The unpruned and pruned DNN models are trained to have nearly equal software accuracies to conduct a fair comparison of the impact of non-idealities when the models are mapped onto non-ideal crossbars (see Table 1). The crossbar-compression-rates for the structure-pruned DNNs on 32×32 crossbars are also shown in Table 1.

Table 1 Table showing software accuracies and crossbar-compression-rates (with 32×32 crossbars) for the various DNN models with CIFAR10 and CIFAR100 datasets

Dataset: CIFAR10	Software accuracy (%) ‖ Crossbar-compression-rate			
Network	Unpruned	C/F ($s = 0.8$)	XCS ($s = 0.8$)	XRS ($s = 0.8$)
VGG11	83.6 ‖ –	83.5 ‖ 19.69×	83.28 ‖ 4.26×	82.67 ‖ 4.88×
VGG16	84.48 ‖ –	83.65 ‖ 19.60×	82.06 ‖ 5.57×	83.47 ‖ 4.89×
Dataset: CIFAR100	Software accuracy (%)			
Network	Unpruned	C/F ($s = 0.6$)		
VGG11	53.29 ‖ –	52.72 ‖ 5.64×		
VGG16	51.83 ‖ –	50.55 ‖ 4.20×		

We find that the sparse DNNs have greatly reduced number of parameters than their unpruned counterparts which results in significantly lesser number of crossbars on hardware. However, the fewer parameters remaining in the sparse DNNs are crucial for the model's performance. Thus, any non-ideality interfering with the fewer parameters of the sparse DNNs would have huge impact on the DNN accuracy and hence, robustness on hardware. In Fig. 18a, we find that for the VGG11/CIFAR10 model, the DNNs with structured sparsity (via C/F pruning, XCS, XRS with $s = 0.8$) suffer greater accuracy degradation than their unpruned counterparts for crossbar sizes ranging from 16×16 to 64×64. Further, as we increase the crossbar size, both the accuracies of unpruned and pruned networks decline owing to increase in crossbar non-idealities [34, 39]. Specifically, on 64×64 crossbars, the inference accuracy of the unpruned model reduces by $\sim$21% with respect to the software baseline while, for the sparse DNNs pruned via C/F pruning, XCS and XRS, the decline is $\sim$39%, $\sim$24% and $\sim$30%, respectively. Also, in Fig. 18b, we find that on reducing the extent of sparsity in the C/F pruned DNNs from $s = 0.8$ to $s = 0.5$, the performance degradation suffered by the pruned DNNs is reduced. This validates the fact that greater sparsity, although leads to energy- and area-efficient mappings on crossbars, increases the interference of crossbar non-idealities, thereby hampering the performance and hence, the robustness of the pruned networks.

In Fig. 18c, for the VGG16 DNN with CIFAR10 dataset, the trends are similar to the case of the VGG11 DNN for XCS, XRS, and C/F pruning ($s = 0.8$) in case of 16×16 and 32×32 crossbars. However, in case of a larger 64×64 crossbar, we find that the performance of the network pruned by C/F pruning exceeds that of the unpruned network. This is because unpruned DNNs require a larger absolute number of crossbars for mapping than pruned ones. As a result, the value of NF is expected to increase at a higher rate for unpruned DNN on moving from 32×32 to 64×64 crossbars (see Fig. 18d). So, for larger crossbars, the accuracy degradation for structure-pruned DNNs would decelerate compared to their unpruned counterparts, which can even lead to better absolute accuracy of the pruned networks than the unpruned ones.

Next, we discuss two crossbar-aware non-ideality mitigation strategies that can help improve the robustness of structure-pruned DNNs on non-ideal crossbars.

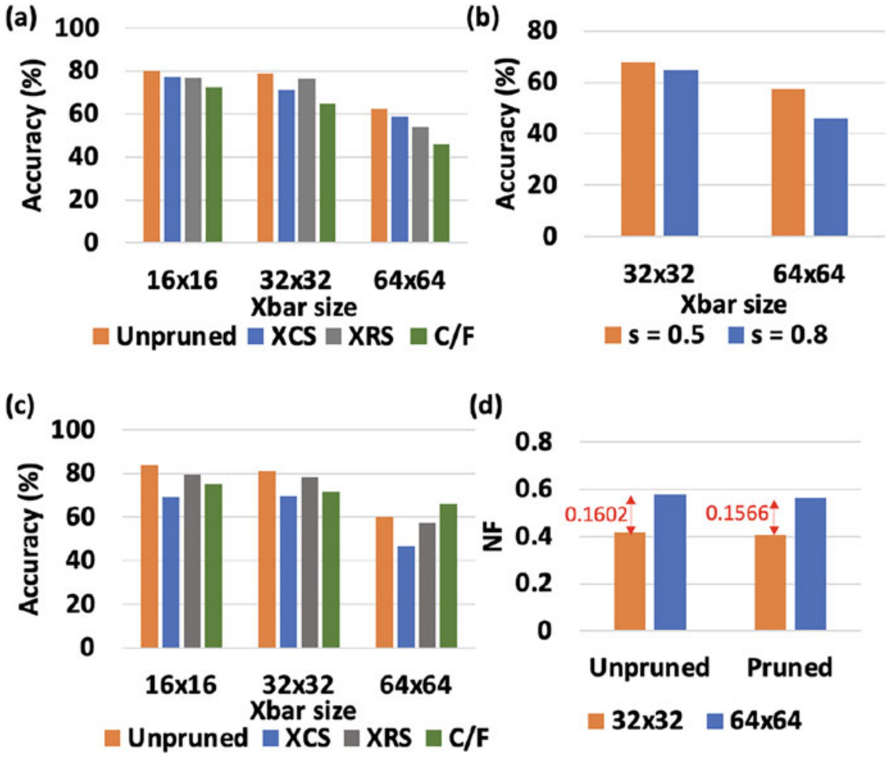


Fig. 18 Plot of inference accuracy versus crossbar size for (**a**) unpruned and structure-pruned ($s = 0.8$) VGG11/CIFAR10 DNN. (**b**) Different values of sparsity (s) of a C/F pruned VGG11/CIFAR10 DNN. (**c**) Unpruned and structure-pruned ($s = 0.8$) VGG16/CIFAR10 DNN. (**d**) Plot showing the variation in average NF for unpruned and C/F pruned weight matrices on increasing the crossbar size from 32×32 to 64×64

5.3 Non-ideality Mitigation Strategies for Increased Robustness of Structure-Pruned DNNs

1. **Crossbar-Column rearrangement** (R)**:** For the sparse DNNs obtained via C/F pruning, a simple hardware-friendly transformation of column rearrangement R has been proposed mapping weights onto non-ideal crossbars. This transformation is inspired from the fact that the impact of non-idealities (or non-ideality factor NF) reduces for crossbars with higher proportion of low conductance synapses [26, 44]. Additionally, this approach of column rearrangement does not have any training overhead and is applied before the mapping of the DNNs onto crossbars.

   To understand column rearrangement R, consider a 4×6 weight matrix W, after applying the transformation T, to be mapped onto 2×2 crossbars (see Fig. 19a). During the R transformation, we first compute the value of $(\mu \times \sigma)^{\frac{1}{2}}$

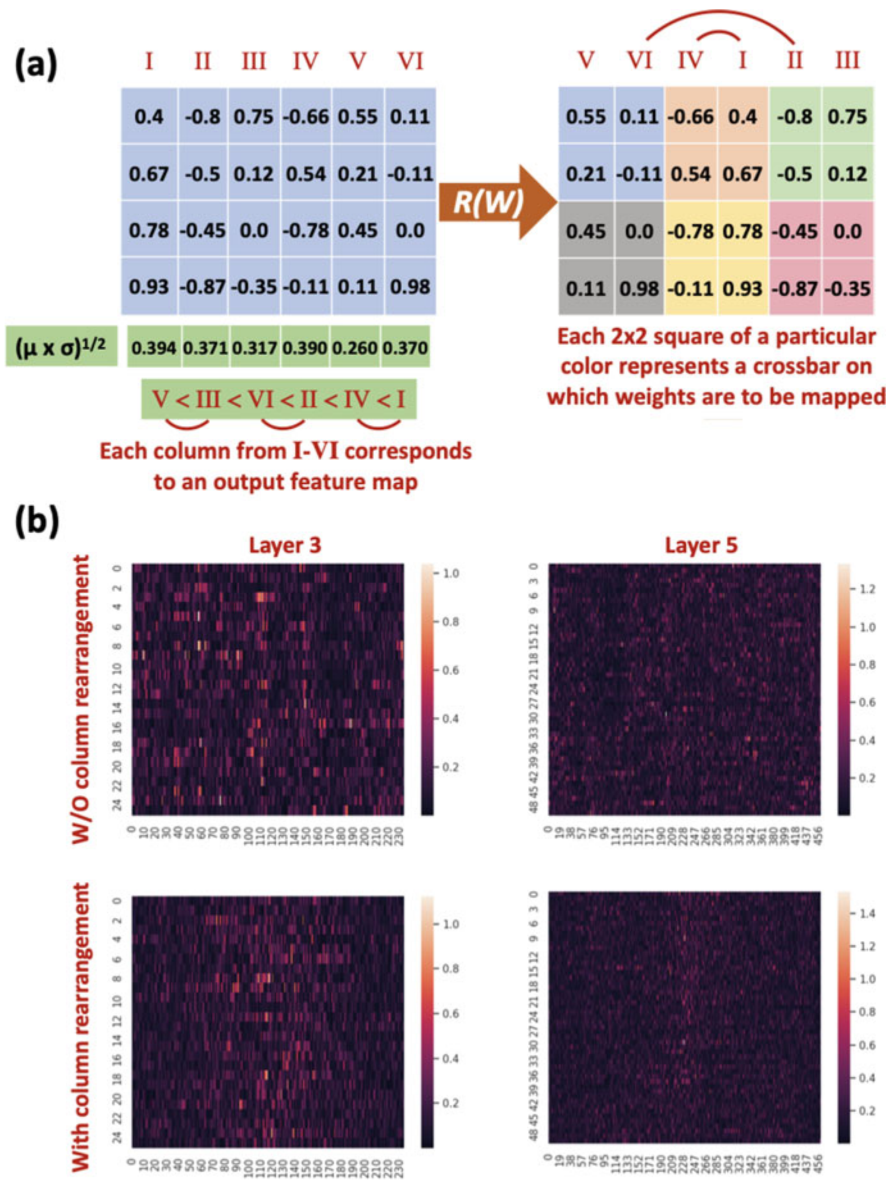


Fig. 19 (**a**) Pictorial representation of R transformation. (**b**) Heatmaps to visualize the impact of R transformation on the weight matrices of 3rd and 5th layers of the VGG16/CIFAR10 DNN trained with C/F pruning with $s = 0.8$

for each column from I–VI, where μ and σ, respectively, denote the mean and standard deviation of the absolute values of weights in each column. Thereafter, based on the increasing order of $(\mu \times \sigma)^{\frac{1}{2}}$, we rearrange columns I–VI in

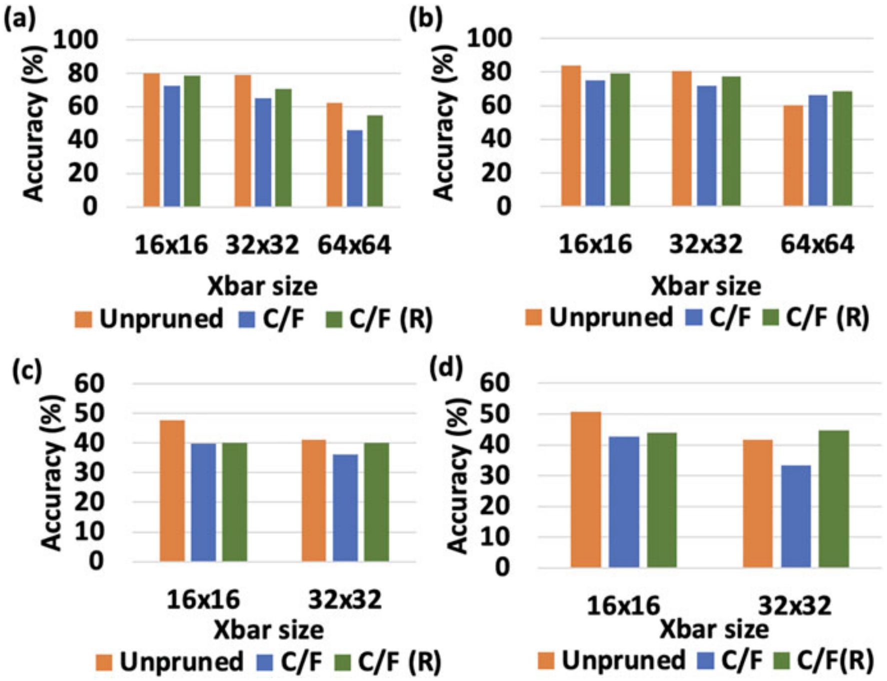


Fig. 20 A plot of inference accuracy versus crossbar size for unpruned, C/F pruned and (**a**) C/F pruned with transformation R ($s = 0.8$) VGG11/CIFAR10 DNNs. (**b**) C/F pruned with transformation R ($s = 0.8$) VGG16/CIFAR10 DNNs. (**c**) C/F pruned with transformation R ($s = 0.6$) VGG11/CIFAR100 DNNs. (**d**) C/F pruned with transformation R ($s = 0.6$) VGG16/CIFAR100 DNNs

the manner shown. Now, in Fig. 19b, the impact of R transformation can be visualized on the weight matrices of the 3rd and 5th convolutional layers of the VGG16/CIFAR10 DNN (C/F pruned with $s = 0.8$) using heatmaps. Before applying the transformation, the lighter (low conductance synapse) and darker (high conductance synapse) points in the heatmaps are intermixed. Post transformation, the lighter points are concentrated at the center of the heatmaps and darker points are mostly near the peripheries. Thus, post R transformation, when the DNN weight matrices are partitioned into multiple crossbar instances, majority of the crossbars have greater proportions of low conductance synapses, thereby mitigating the impact of crossbar non-idealities.

Figure 20a,b and c,d show that R transformation improves the performance of the C/F pruned VGG11 and VGG16 DNNs. Specifically, $\sim$9% ($\sim$6%) improvement in accuracy is observed for VGG11 (VGG16) DNN on 64×64 (32×32) crossbars with CIFAR10 dataset. We also find that on 32×32 crossbars, the accuracy of the pruned VGG16/CIFAR100 DNN post R transformation is $\sim$3% greater than the unpruned counterpart.

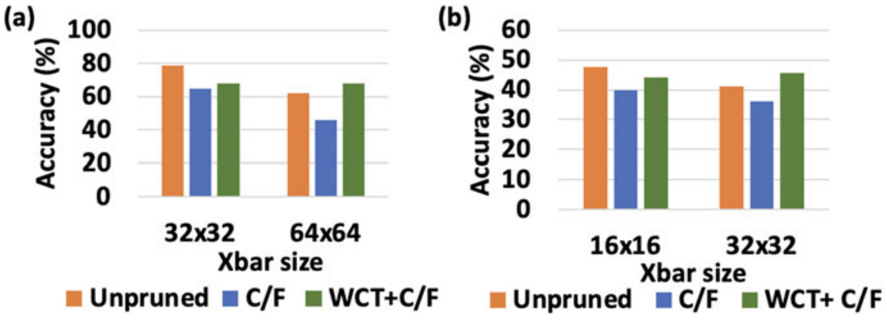


Fig. 21 A plot of inference accuracy versus crossbar size for unpruned, C/F pruned and (**a**) WCT + C/F pruned ($s = 0.8$) VGG11/CIFAR10 DNNs. (**b**) WCT + C/F pruned ($s = 0.6$) VGG11/CIFAR100 DNNs

2. **Weight-Constrained-Training (WCT):** WCT is another non-ideality mitigation technique for the structure-pruned DNNs that is motivated by the NEAT method described in Sect. 3. In WCT, based on the weight distribution of all the layers of a trained DNN, a cut-off value W_{cut} is heuristically determined, and the following transformation is then applied on the weights of the DNN: $W = min\{|W|, W_{cut})\} * sign(W)$. This transformation constrains the DNN weights in the interval $[-W_{cut}, W_{cut}]$. With the above transformation, the DNN is iteratively trained for 1–2 epochs, to maintain nearly iso-accuracy with baseline. Note, the iterative training via WCT does not add any computational overhead to the overall training time, thereby making it a viable choice. Similar to NEAT, a WCT-DNN also results in greater proportion of low conductance states on the crossbars, thus, reducing the impact of non-idealities. The resultant sparse WCT-DNNs are then mapped onto crossbars. In Fig. 21a,b, we find that the WCT-DNNs maintain their performance even on increasing the crossbar size, making them robust against crossbar non-idealities. Further, WCT-DNNs have better accuracy than the C/F pruned DNNs on crossbars. Specifically, with CIFAR10 (CIFAR100) dataset, the WCT-DNN has ~6% (~7%) higher accuracy than the unpruned model on 64×64 (32×32) crossbars.

6 Related Works

Recently, several crossbar-based In-Memory Computation (IMC) architectures and frameworks have been proposed for efficiency-driven acceleration of DNNs [50–54]. CONV-SRAM [50] proposes an energy-efficient static random access memory (SRAM) with embedded dot-product computation capability, for the inference of convolutional neural networks with binary weights. On the other hand, Kim et al. [52] and Gokmen et al. [53] have proposed an architecture based on CMOS-based resistive processing unit (RPU) devices to achieve significant acceleration in DNN training.

In ISAAC [51], Shafiee et al. designed and characterized a pipelined memristive crossbar architecture and proposed a weight encoding scheme that reduces the analog-to-digital conversion overheads. Additionally, Marinella et al. [54] implement an ReRAM crossbar-based DNN acceleration platform and characterize the energy, latency, and area of the peripheral and crossbar components across different technology nodes. Besides crossbar-based DNN acceleration platforms, end-to-end hardware evaluation platforms such as Neurosim and PUMA [37, 49] provide software-based scalable solutions to perform hardware evaluation of crossbar implementations. While Neurosim [49] considers only NVM device variations during DNN evaluation, PUMA [37] models other circuit-level and data-dependent non-idealities by incorporating GenieX [34]. Few recent works such as RxNN [23] and GenieX [34] have delved deeper into modelling the characteristics of non-ideal crossbars. These non-idealities include crossbar-interconnect parasitics and data-dependent selector non-linearities. While RxNN can suitably compute data-independent non-idealities, GenieX incorporates both data-dependent and independent crossbar non-idealities. With this, they provide accurate hardware-realistic inference performance of crossbar-mapped DNNs.

However, none of these works has explored non-ideality aware crossbar-mapping of DNN models for adversarial robustness. Furthermore, these works have not delved into the correlation existing between sparsity in network weights and crossbar non-idealities to highlight the vulnerabilities of sparse DNNs. Additionally in prior works, the possibility of using inherent hardware signatures in the detection of adversarial attacks and building adversarial robustness for crossbar-mapped DNN models has not been well explored. This motivates us to present recent works involving non-ideality aware mapping of DNNs onto crossbars for improving their classification accuracy (robustness) in normal and/or adversarial scenarios [26, 33]. In addition, examining hardware signatures in crossbars for energy-efficient adversarial detection [27] is a key facet of this chapter. We present a summary table (Table 2) for the convenience of the readers to qualitatively compare the scope of different works based on memristive crossbar arrays pertaining to DNN inference acceleration.

7 Conclusion

This chapter elucidates recent advances in the energy-efficient and robust implementation of DNNs on memristive crossbar array platforms. Specifically, we come across works that use hardware-driven methods to improve the adversarial security of DNNs on noisy crossbars without additional overhead of retraining or reduced energy-efficiency. The first work (NEAT) improves the adversarial classification capabilities on DNNs on crossbars, while the other work (DetectX) is an adversarial detection method to guarantee robustness on crossbar platforms. Additionally, this

Table 2 Table comparing the scope of different memristive crossbar-based works for DNNs. The works discussed in this chapter—DetectX [27], NEAT [26], and Bhattacharjee et al. [33] have specifically added a new dimension of adversarial and sparsity-aware robustness which have not been looked into in prior works

	Xbar acceleration		End-to-end H/W	Robustness		
Work	Efficiency-driven	Novel weight mapping	Evaluation	Sparsity-aware	Non-ideality	Adversarial attacks
CONV-SRAM [50]	✓	×	×	×	×	×
ISAAC [51]						
Kim et al. [52]						
Gokmen et al. [53]						
Marinella et al. [54]						
Neurosim [49]	✓	×	✓	×	✓	×
PUMA [37]						
RxNN [23]	✓	×	×	×	✓	×
GenieX [34]						
DetectX [27]	✓	×	×	×	✓	✓
NEAT [26]	✓	✓	×	×	✓	✓
Bhattacharjee et al. [33]	✓	✓	×	✓	✓	×

chapter highlights a study which corroborates that although increased structured sparsity in weights is beneficial for resource-efficient implementation of DNNs on crossbars, it compromises their classification accuracy (robustness) in a non-ideal scenario. To this end, various hardware-based non-ideality mitigation approaches have been proposed to improve the performance and hence, the robustness of sparse DNNs on crossbars.

References

1. Reagen, B., et al.: Minerva: enabling low-power, highly-accurate deep neural network accelerators. In: 2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA), pp. 267–278. IEEE, Piscataway (2016)
2. Hadidi, R., et al.: Characterizing the deployment of deep neural networks on commercial edge devices. In: 2019 IEEE International Symposium on Workload Characterization (IISWC), pp. 35–48. IEEE, Piscataway (2019)
3. Chakraborty, I., et al.: Pathways to efficient neuromorphic computing with non-volatile memory technologies. Appl. Phys. Rev. (2020)
4. Wong, H.-S.P., et al.: Metal–oxide RRAM. Proc. IEEE **100**(6), 1951–1970 (2012)
5. Dodge, S., Karam, L.: A study and comparison of human and deep learning recognition performance under visual distortions. In: 2017 26th International Conference on Computer Communication and Networks (ICCCN), pp. 1–7. IEEE, Piscataway (2017)

6. Kurakin, A., Goodfellow, I., Bengio, S.: Adversarial machine learning at scale (2016). Preprint. arXiv:1611.01236
7. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples (2014). Preprint. arXiv:1412.6572
8. Madry, A., et al.: Towards deep learning models resistant to adversarial attacks (2017). Preprint. arXiv:1706.06083
9. Carlini, N., et al.: On evaluating adversarial robustness (2019). Preprint. arXiv:1902.06705
10. Lin, J., Gan, C., Han, S.: Defensive quantization: when efficiency meets robustness (2019). Preprint. arXiv:1904.08444
11. Qiu, H., et al.: Mitigating advanced adversarial attacks with more advanced gradient obfuscation techniques (2020). Preprint. arXiv:2005.13712
12. Guo, C., et al.: Countering adversarial images using input transformations (2017). Preprint. arXiv:1711.00117
13. Prakash, A., et al.: Deflecting adversarial attacks with pixel deflection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 8571–8580 (2018)
14. Xie, C., et al.: Mitigating adversarial effects through randomization (2017). Preprint. arXiv:1711.01991
15. Buckman, J., et al.: Thermometer encoding: one hot way to resist adversarial examples. In: International Conference on Learning Representations (2018)
16. Metzen, J.H., et al.: On detecting adversarial perturbations (2017). Preprint. arXiv:1702.04267
17. Yin, X., Kolouri, S., Rohde, G.K.: Gat: generative adversarial training for adversarial example detection and robust classification. In: International Conference on Learning Representations (2019)
18. Sterneck, R., Moitra, A., Panda, P.: Noise sensitivity-based energy efficient and robust adversary detection in neural networks. In: IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (2021)
19. Panda, P., Chakraborty, I., Roy, K.: Discretization based solutions for secure machine learning against adversarial attacks. IEEE Access **7**, 70157–70168 (2019)
20. Gui, S., et al.: Model compression with adversarial robustness: a unified optimization framework. In: Wallach, H., et al. (eds.) Advances in Neural Information Processing Systems, vol. 32. Curran Associates Inc., Red Hook (2019)
21. Sehwag, V., et al.: Hydra: pruning adversarially robust neural networks. Adv. Neural Inf. Proces. Syst. **33**, 19655–19666 (2020)
22. Panda, P.: QUANOS-adversarial noise sensitivity driven hybrid quantization of neural networks. Preprint. arXiv:2004.11233 (2020)
23. Jain, S., et al.: RxNN: a framework for evaluating deep neural networks on resistive crossbars. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. (2020)
24. Bhattacharjee, A., Panda, P.: Rethinking non-idealities in memristive crossbars for adversarial robustness in neural networks (2020). Preprint. arXiv:2008.11298
25. Roy, D., et al.: Robustness hidden in plain sight: can analog computing defend against adversarial attacks? (2020). arXiv: 2008.1201
26. Bhattacharjee, A., et al.: NEAT: non-linearity aware training for accurate, energy-efficient and robust implementation of neural networks on 1T-1R crossbars. In: IEEE Transactions on Computer-Aided Design (2021)
27. Moitra, A., et al.: DetectX – adversarial input detection using current signatures in memristive XBar arrays. In: IEEE Transactions on Circuits and Systems I (2021)
28. Wen, W., et al.: Learning structured sparsity in deep neural networks (2016). Preprint. arXiv:1608.03665
29. Wang, P., et al.: SNrram: an efficient sparse neural network computation architecture based on resistive random-access memory. In: 2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC) (2018)
30. Liang, L., et al.: Crossbar-aware neural network pruning. IEEE Access (2018)

31. Lin, J., et al.: Learning the sparsity for ReRAM: mapping and pruning sparse neural network for ReRAM based accelerator. In: ASPDAC '19: Proceedings of the 24th Asia and South Pacific Design Automation Conference (2019)
32. Chu, C., et al.: PIM-prune: fine-grain DCNN pruning for crossbar-based process-in-memory architecture. In: 2020 57th ACM/IEEE Design Automation Conference (DAC) (2020)
33. Bhattacharjee, A., Bhatnagar, L., Panda, P.: Examining and mitigating the impact of crossbar non-idealities for accurate implementation of sparse deep neural networks. In: 2022 Design, Automation & Test in Europe Conference & Exhibition (DATE) (2022)
34. Chakraborty, I., et al.: GENIEx: a generalized approach to emulating non-ideality in memristive Xbars using neural networks (2020). Preprint. arXiv:2003.06902
35. Liu, B., et al.: Vortex: variation-aware training for memristor x-bar. In: Proceedings of the 52nd Annual Design Automation Conference, pp. 1–6 (2015)
36. Lee, S., et al.: Learning to predict IR drop with effective training for ReRAM-based neural network hardware. In: 2020 57th ACM/IEEE Design Automation Conference (DAC), pp. 1–6. IEEE, Piscataway (2020)
37. Ankit, A., 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, pp. 715–731 (2019)
38. Ansari, M., et al.: PHAX: physical characteristics aware ex-situ training framework for inverter-based memristive neuromorphic circuits. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **37**(8), 1602–1613 (2017)
39. Bhattacharjee, A., Moitra, A., Panda, P., Efficiency-driven hardware optimization for adversarially robust neural networks. In: Design, Automation and Test in Europe Conference (DATE) (2021)
40. Chen, P.-Y., et al.: Mitigating effects of non-ideal synaptic device characteristics for on-chip learning. In: 2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), pp. 194–199. IEEE, Piscataway (2015)
41. Agrawal, A., Lee, C., Roy, K.: X-CHANGR: changing memristive crossbar mapping for mitigating line-resistance induced accuracy degradation in deep neural networks (2019). Preprint. arXiv:1907.00285
42. Liu, B., et al.: Reduction and IR-drop compensations techniques for reliable neuromorphic computing systems. In: 2014 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), pp. 63–70. IEEE, Piscataway (2014)
43. He, Z., et al.: Noise injection adaption: end-to-end ReRAM crossbar nonideal effect adaption for neural network mapping. In: Proceedings of the 56th Annual Design Automation Conference 2019, pp. 1–6 (2019)
44. Bhattacharjee, A., et al.: SwitchX: gmin-gmax Switching for energy-efficient and robust implementation of binary neural networks on ReRAM Xbars (2021). Preprint. arXiv:2011.14498
45. Sun, X., Yu, S.: Impact of non-ideal characteristics of resistive synaptic devices on implementing convolutional neural networks. IEEE J. Emerging Sel. Top. Circuits Syst. **9**(3), 570–579 (2019)
46. Li, T., et al.: Sneak-path based test and diagnosis for 1r RRAM crossbar using voltage bias technique. In: Proceedings of the 54th Annual Design Automation Conference 2017, pp. 1–6 (2017)
47. Wang, Z., et al.: Ferroelectric tunnel memristor-based neuromorphic network with 1T1R crossbar architecture. In: 2014 International Joint Conference on Neural Networks (IJCNN), pp. 29–34. IEEE, Piscataway (2014)
48. He, Z., Rakin, A.S., Fan, D.: Parametric noise injection: trainable randomness to improve deep neural network robustness against adversarial attack. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 588–597 (2019)
49. Chen, P.-Y., Peng, X., Yu, S.: NeuroSim: a circuit-level macro model for benchmarking neuro-inspired architectures in online learning. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **37**(12), 3067–3080 (2018)

50. Biswas, A., Chandrakasan, A.P.: CONV-SRAM: an energy-efficient SRAM with in-memory dot-product computation for low-power convolutional neural networks. IEEE J. Solid State Circuits **54**(1), 217–230 (2018)
51. Shafiee, A., et al.: ISAAC: a convolutional neural network accelerator with in-situ analog arithmetic in crossbars. ACM SIGARCH Comput. Archit. News **44**(3), 14–26 (2016)
52. Kim, S., et al.: Analog CMOS-based resistive processing unit for deep neural network training. In: 2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS), pp. 422–425. IEEE, Piscataway (2017)
53. Gokmen, T., Vlasov, Y., Acceleration of deep neural network training with resistive cross-point devices: design considerations. Front. Neurosci. **10**, 333 (2016)
54. Marinella, M.J., et al.: Multiscale co-design analysis of energy, latency, area, and accuracy of a ReRAM analog neural training accelerator. IEEE J. Emerging Sel. Top. Circuits Syst. **8**(1), 86–101 (2018)