Lower Voltage for Higher Security: Using Voltage Overscaling to Secure Deep Neural Networks

Md Shohidul Islam*§, Ihsen Alouani‡, Khaled N. Khasawneh*
*ECE Dept., George Mason University, Fairfax, VA, USA
§CSE Dept., Dhaka University of Engineering & Technology, Gazipur, Bangladesh
‡Queen's University Belfast
Email: {mislam20, kkhasawn}@gmu.edu, ihsen.alouani@uphf.fr

***Abstract*—Deep neural networks (DNNs) are shown to be vulnerable to adversarial attacks—— carefully crafted additive noise that undermines DNNs integrity. Previously proposed defenses against these attacks require substantial overheads, making it challenging to deploy these solutions in power and computational resource-constrained devices, such as embedded systems and the Edge. In this paper, we explore the use of voltage overscaling (VOS) as a lightweight defense against adversarial attacks. Specifically, we exploit the stochastic timing violations of VOS to implement a moving-target defense for DNNs. Our experimental results demonstrate that VOS guarantees effective defense against different attack methods, does not require any software/hardware modifications, and offers a by-product reduction in power consumption.**

***Index Terms*—Adversarial attacks, machine learning, approximate computing, voltage overscaling**

I. Introduction and Related Work

In the last few years, an increasing number of deep learning architectures, such as Convolutional Neural Networks (CNNs), aka Deep Neural Networks (DNNs), have been deployed to tackling a wide range of complex real-life problems. This widespread usage covers multifarious applications: Vision, robotics, speech recognition, natural language processing, financial fraud detection, malware detection, etc. Owing to their proven performance, CNNs have been deployed even in safety-critical applications such as autonomous vehicles [1] and security-sensitive applications such as automatic bank cheque processing [2]. While CNNs development is progressing rapidly, they are shown vulnerable to adversarial attacks—small perturbations into the input samples can completely fool the classifier and generate wrong output labels. The consequence of such misprediction can be dramatic. For example, in self-driving cars, misclassification of *stop* sign as *yield* sign or *speed limit* sign can claim life or material damage. Until recently, a number of adversarial attacks have been demonstrated in real-world scenario [3], [4] that pose an alarming threat to the safety and security aspect of CNN-powered applications. We use the terms 'CNNs' and 'DNNs' interchangeably in the paper. Furthermore, even machine learning based malware detectors are vulnerable to adversarial attacks [5]–[9].

In the quest for robust DNNs, several defense techniques have been proposed in the literature that can be grouped into four main categories as follows: Adversarial Training (AT) [10]–[12], Input Preprocessing [13]–[15], Gradient Masking (GM) [16], and Randomization-based Defenses [17]–[20]. AT proposed to retrain the models with adversarial inputs in order to detect similar attacks during test time; the technique works perfectly for known attacks; however, it fails to detect unknown types of attacks. AT also comes with other challenges such as generating adversarial samples are compute-intensive and retraining the models (with adversarial inputs) is time-intensive, making the solution inefficient. Input preprocessing based defenses perform some transformations to the inputs that, in essence, nullify the effect of adversarial perturbations; the technique functions in a limited settings but is highly susceptible to white-box attacks where attackers have access to model's gradients and the preprocessing units. GM is model regularization based defense which requires retraining, and more importantly, the defense has been undermined by the renowned Carlini & Wagner attack [21]. Randomization based defenses introduce random noise in the entire DNNs [19] or in some select layers [17], which stochastically changes the classifier's decision boundary, making it a moving target defense. Such techniques have shown theoretical guarantee of robustness, but the existing solutions are not implemented at scale (e.g., Raghunathan et al. [20] evaluate only a tiny neural network.), nor did they provide any practical source of random noise. It is important to note that our proposed defense is closest to the randomization based technique, where we overcome the aforementioned limitations.

Inspired by [17], [18], our work aims at injecting noise into the CNN computations to defend against adversarial attacks. Instead of theoretical noise distributions that are impractical, we propose a new paradigm in which we leverage stochastic, hardware-induced noise to enhance DNNs robustness. Specifically, we unprecedentedly propose to use VOS as a defense against adversarial attacks. VOS is originally explored in the approximate computing paradigm to reduce computational complexity and power consumption at the expense of accuracy. This paper leverages it for a *totally new objective*, namely DNNs robustness, where power saving is a by-product gain. Our defense exploits the stochastic noise injected by VOS-induced random timing violations. This stochastic behavior makes the technique a practical moving-target defense; making the gradient direction estimation very challenging for an adversary, even with full knowledge of the target model and the defense mechanism. The contributions of this paper are

summarized as follows.

1) We perform the characterization of VOS-induced computational faults properties on a real CPU. Our results show that VOS faults are stochastic (i.e., time-variant) and controllable.
2) We unprecedentedly leverage VOS as a robustness enhancement technique for DNNs; we show that hardware-induced noise can be utilized for securing DNNs in a practical, easy-to-deploy, and power-efficient manner.
3) With no retraining overhead and by-product gain in power savings, our method shows promising results under strong white-box and black-box attack settings.
4) For research reproducibility and to encourage the community to further explore this technique, our code is open [1].

II. Background

A. Adversarial Attacks

Adversarial examples consist of small perturbations added to the inputs that can completely fool the victim models. Considering an example in computer vision, adversarial inputs are generated by adding visually imperceptible noise to the original image, which can mislead the classifier and produce wrong labels. To state it formally, let $h(\cdot)$ be a DNN model used for $m-ary$ image classification, x an original image assigned to ground truth class $c = h(x)$, where $c \in \{1, 2, ..., m\}$. The adversarial attack's goal is to add low amplitude noise to x to generate a perturbed image x' that has a different label from c. The adversarial example can be formally expressed by the following constrained optimization [22]:

$$\begin{aligned} x' = \arg\min_{x'} \quad & \mathcal{D}(x, x'), \\ s.t. \quad & c' = h(x'), \quad c' \in \{1, 2, ..., m\} \setminus \{c\}, \\ & \mathcal{D}(x, x') \leq \varepsilon, \end{aligned} \tag{1}$$

Where, ε is the noise budget and $\mathcal{D}$ is the visual difference between x and x', which is measured using $\ell_p - norm$ as follows:

$$\mathcal{D}(x, x') = \left(\sum_{i=1}^{n} |\Delta x_i|^p \right)^{\frac{1}{p}}; \quad p \in \{0, 2, \infty\} \tag{2}$$

Where, ℓ_0 counts the number of pixels with different values at corresponding locations, ℓ_2 measures the Euclidean distance and ℓ_∞ is the Chebyshev distance measuring the maximum difference for all pixels at corresponding locations. Given a set of N original images $\{x_1, x_2, ..., x_N\}$ and their respective perturbed images $\{x'_1, x'_2, ..., x'_N\}$, the success rate of adversarial attack is measured as follows:

$$Attack\ success\ rate = \frac{1}{N} \sum_{i=1}^{N} 1\left[h\left(x'_i\right) \neq h\left(x_i\right) \right] \tag{3}$$

[1] https://github.com/CAMLsec/DNNs-Undervolting

B. VOS Basics

To cope with the end of Moore's Law and performance requirements of emerging applications, VOS has been used in error tolerant applications; it consists of reducing supply voltage without adapting the operating frequency. In this section, we discuss the impact of VOS on hardware behavior.

While transistor size shrinks with the new generation technologies, the effect of process variation becomes more critical from a digital design perspective. In fact, lower device dimensions sharpen the circuit sensitivity to variations such as imperfection of the manufacturing process, random dopant fluctuation, and variation in the gate oxide thickness. With reduced transistors dimensions, the standard deviation of threshold voltage variation ($\Delta \mathrm{V_t}$) increases since it is proportional to the square root of the device area [23]:

$$\sigma_{\Delta \mathrm{V_t}} = \frac{\mathrm{A}_{\Delta \mathrm{V_t}}}{\sqrt{\mathrm{WL}}} \tag{4}$$

where W and L are the width and the length of the device, respectively, and $\mathrm{A}_{\Delta \mathrm{V_t}}$ is characterizing matching parameter for any given process. This variation in V_t has a direct impact on the circuit delay, which can be approximated using the following equation [23]:

$$d_{gate} \propto \frac{V_{DD}}{\beta \left(V_{DD} - V_t\right)^{\alpha}} \tag{5}$$

where α and β are fitting parameters for a given gate in a given process. For this reason, in the circuit design phase, static timing analysis is generally achieved to verify that all circuit paths meet the timing requirements to produce correct output regardless of the input combination under given supply voltage. Scaling down the supply voltage (V_{DD}) from the nominal operating voltage results in slowing down signals propagation and thereby creating a timing overhead. If this process is *not* accompanied with a corresponding frequency scaling, timing errors may occur within the circuit results: this is the case of VOS.

III. Characterization of VOS-induced faults

After explaining the basis of VOS induces computational faults (Section II-B), we empirically characterize these faults on a real CPU. The characterization goal is to understand the fault model of VOS, and verify if the VOS computational fault have the properties that qualify it to be used as a randomization based defense for securing DNNs. In particular, we are interested to know if the VOS-induced faults are: (1) stochastic: to obfuscate the classifier behavior, and (2) controllable: to allow balancing the security and accuracy trade-offs.

In our experiment, we used an Intel Broadwell processor (model number i7-5557U) running on Ubuntu 16.04 LTS with stock Linux v4.15. We used the model specific register (MSR) to control the voltage of the CPU from software [24]–[26]. In particular, we dynamically scaled voltage through `MSR 0x150` (a 64-bit register) where 3-bit (42-40) `plain idx` locates the CPU components to apply voltage, and 11-bit (31-21) `offset` indicates the requested voltage scaling offset. Encoding the

TABLE I: Examples of faulted multiplication on i7-5557U at 2.2 GHz. Figures in the table are in Hexadecimal format. Op1 and Op2 are the operands of multiplication. Red color in the VOS result indicates fault location.

Op1	Op2	Op1 × Op2 VOS result	Op1 × Op2 Golden result
0x59a2f277	0xbee4b58	0x042d709da4ae35e8	0x042d704a14ae35e8
0x59a2f277	0xbee4b58	0x0e2d704a14a205e8	0x042d704a14ae35e8
0x59a2f277	0xbee4b58	0x042d704a141735e8	0x042d704a14ae35e8
0x2d18c998	0x749ffff	0x014844ad40d73668	0x0148b6ad40d73668
0x2d18c998	0x749ffff	0x0148b6ad40dc3668	0x0148b6ad40d73668
0x2d18c998	0x749ffff	0x0148b6ade5d73668	0x0148b6ad40d73668
0xffffffff	0x6c4931e	0x0664931df93b6ae2	0x06c4931df93b6ce2
0xffffffff	0x6c4931e	0x06c493adf93b6ce2	0x06c4931df93b6ce2
0xffffffff	0x6c4931e	0x06c4931d103b6ce2	0x06c4931df93b6ce2

voltage offset (with respect to core's base operating voltage) using 11-bit signed integer allows us to achieve a step size of $^1/_{1024}$ V (about 1 mV), thus allowing a maximum voltage offset of ± 1 V. We set `plain idx` to `0` to adjust the voltage of 'processor core' and kept CPU frequency at 2.2 GHz. We observed that too little reduction in voltage does not produce any fault but too far reduction results in system freezes or crashes. Thus, to better understand the nature of the faults, we reduced voltage by a small step size of 1 mV while repeatedly executing the same instruction with the same operands until a fault or system freeze occurs. Below, we will show the analysis of VOS effect on multiplications as well as other operations.

A. VOS effect on multiplication

To study VOS effect on multiplication, we conducted an experiment where a multiplication operation with the same operands is executed multiple times. We repeat this experiment for different sets of operands. Table I shows some selected results, i.e., for faulty multiplications, from our experiments. We noticed that reducing the voltage by -103 mV to -145 mV, depending on inputs, was sufficient to generate faults. In addition, the sign bit of the output was never flipped; the output sign bit is a simple XOR operation of the operand's sign bits, which is far from the critical path of the multiplier circuit. Moreover, the fault occurred as one to eight (contiguous) bits and the 8 least significant bits of the output were never flipped, mainly due to their small propagation delay. Notice in Table I (column 'VOS result') that the fault locations for the same operands varied non-deterministically. More interestingly, we found that the pair of operands generating faults in one run sometimes generate correct results in another run. Thus, the nature of VOS-induced faults are practically stochastic.

Although from circuit perspective, under a given voltage, for the same dopant fluctuation, and thermal condition, the timing violation should be systematic, the stochastic faults behavior is because of the critical path being input dependent. In other words, different sets of operands may lead to different critical-path lengths for a given operation. In addition, the temperature's impact on voltage threshold and delay varies with the voltage [27], [28]. Hence, by considering on-chip thermal variability, timing violations are stochastically impacted by temperature.

After empirically observing that the VOS-induced faults are stochastic, we check whether the faults are controllable. Therefore, we setup an experiment such that multiplications are repeatedly executed until a fault occurs. Figure 1 shows the number of iterations, i.e., multiplications, needed to observe the first fault for different voltage offsets, i.e, while scaling the voltage down. The result shows that nominal reduction in voltage requires higher number of iterations while aggressive voltage reduction requires fewer number of iterations; the probability of computational faults decreases while scaling the voltage up and vice versa. Thus, demonstrating that VOS induced faults are controllable.

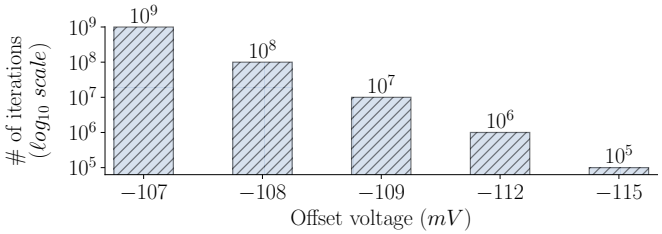


Fig. 1: Number of multiplications required to observe a single computational fault vs. necessary voltage offset from $nominal - V_{dd}$ on i7-5557U at 2.2 GHz

B. VOS effect on other operations

Likewise multiplication instructions, we conducted the same experiments on other instructions including *addition, subtraction, bit-wise AND, OR,* and *Shift* operations. However, the results did not show any faults for these operations while scaling the voltage down. The reason for not having faulty results for those operations is that their circuit implementation is simple, thus, requiring smaller propagation delay compared to multiplication operations.

IV. Threat Model

In this section, we explain the threat model against which the robustness of DNN models is evaluated. For this purpose, we consider two attack scenarios such as Black-box attack and White-box attack as described below.

Black-box attack. In this setting, we assume the attacker has access only to the input/output of the victim classifier (which is our defender classifier) and has no information about its internal architecture. For this family of attacks, we use *HopSkipJump (HSJ)* [29], which is one of the most recent and powerful black-box attacks. HSJ is a decision-based iterative attack that does not require the victim model's gradients. Instead, it relies on the final output of the target model, estimates the gradients, and approximates gradient direction based on the binary information at the decision boundary.

White-box attack. In white-box setting, the attacker has total access to the victim's internal model. This includes inputs, outputs, model architecture, parameters and hyper-parameters, gradient, etc. Notice that since this attack model does not

consider any constraints on the attacker's knowledge nor access, it is more difficult to defend against. The attacker's goal is to utilize the knowledge of the internal model of the victim CNN to systematically create adversarial attacks. For this setting, we use Carlini & Wagner ($C\&W$) attack [21] and Projected gradient descent (PGD) attack [30].

(1) $C\&W$: It is one of the most powerful state-of-the-art gradient-based adversarial attacks, which generates adversarial samples based on the following optimization problem:

$$\underset{\delta}{\text{minimize}} \quad \|\delta\|_2 + c \cdot \ell(x+\delta) \ s.t. \quad x+\delta \in [0,1]^n$$

Where $\|\delta\|_2$ is the ℓ_2 measure of the smallest perturbation that can change the model's prediction output and $\ell(\cdot)$ is the loss function, which captures the difference between current iteration and the objective of the attack as defined below:

$$\ell(x) = max(max_{i \neq t}\{Z(x)\} - Z(x)_t - \kappa)$$

Where $Z(x)$ is the logits before applying softmax function, t is the target label, and κ is the class confidence.

(2) PGD: It is the strongest iterative variant of Fast Gradient Sign Method (FGSM) attack [10], where the adversarial example is generated as follows.

$$x^{t+1} = \mathcal{P}_{S_x}(x^t + \alpha \cdot sign(\nabla_x \mathcal{L}_\theta(x^t, y)))$$

Where $\mathcal{L}(\cdot)$ is the loss function, θ is the set of model parameters, ∇ is the gradient of loss, $\mathcal{P}_{S_x}$ is a projection operator projecting the input into the feasible region S_x and α is the amount of added noise in each iteration. Over the iterations, PGD attack tries to find the perturbation that essentially maximizes the loss on a given input while keeping the size of perturbation smaller than the specified amount. We summarize these attack methods in Table II.

TABLE II: Summary of the used attack methods. Notice that the strength estimation (out of 5 stars) is based on [31].

Method	Category	Perturb. Norm	Learning	Strength
C&W	Gradient-based	ℓ_2, ℓ_∞	Iterative	*****
PGD	Gradient-based	ℓ_2, ℓ_∞	Iterative	****
HSJ	Decision-based	ℓ_2, ℓ_∞	Iterative	*****

V. Proposed Approach

In this work, we use hardware-induced noise, specifically through VOS, to defend DNNs against adversarial attacks. The rationale behind choosing VOS is as follows:

(i) It injects stochastic behavior: Injecting random noise's positive impact on DNNs robustness has been proven theoretically in [17], [18]. However, none of the related work has shown a practical and controllable source of randomness that does not require high overhead and considerable complexity. One of the fundamental properties of VOS is that the induced timing violations are stochastic, as explained in Section III. We leverage these properties as a natural and easy to deploy source of random noise inside convolution layers.

(ii) Stochastic noise is controllable: While injected random noise can be used to improve the robustness of DNNs [17], [18], if the injected noise can't be controlled, i.e., bounded, the noise would drastically decrease the classifier accuracy on clean input, i.e., not adversarial, as we show later in Figure 9. Nonetheless, as we showed in Section III, the VOS induced computational faults are controllable, which enables us to balance the accuracy and security trade-offs.

(iii) Easy to deploy: VOS deployment does not include any specific changes to the underlying hardware nor to the running software, except for varying the supply voltage. It is also model-agnostic and does not require retraining the model, nor does it require fine tuning parameters or changing hyper-parameters. These properties make it highly practical and represent a drop-in solution, contrasting with prior techniques, which require high overheads. Moreover, VOS can be used on DNNs hardware accelerators in System-on-Chips without impacting other components reliability, especially if they are not fault-tolerant.

(iv) It reduces energy consumption: The very essence of VOS is reducing supply voltage without adapting frequency. It thereby comes with a high energy consumption reduction because of the super-linear dependence of both dynamic and leakage power on the supply voltage as follows:

$$P = \alpha C V_{DD}^2 f + V_{DD} I_{leakage} \tag{6}$$

Where α is the activity factor, C is the total capacitive load, f is the frequency, $I_{leakage}$ is the leakage current, V_{DD} is the supply voltage, and P is the total power consumption.

To model the impact of VOS on a large-scale system such as a DNN, we utilized the VOS induced computational faults characterization results shown in Section III. In particular, based on the computational fault model generated through our characterization, stochastic errors will be injected in multiply-accumulate (MAC) computations. To implement this, we modify the PyTorch platform to integrate MAC behavior under a given fault probability, thereby simulating convolution layers under VOS. We simulate the impact of VOS on CNNs inference by selecting the corresponding timing error rate to be injected at runtime.

> ***Terminology.*** In the remainder of the paper, we call *exact model*: a conventional trained CNN model and *approximate model*: a model where we apply VOS at inference.

VI. Security Evaluation

This section first describes the experimental setup. Subsequently, we present the robustness evaluation of the proposed defense against both black-box and white-box attacks.

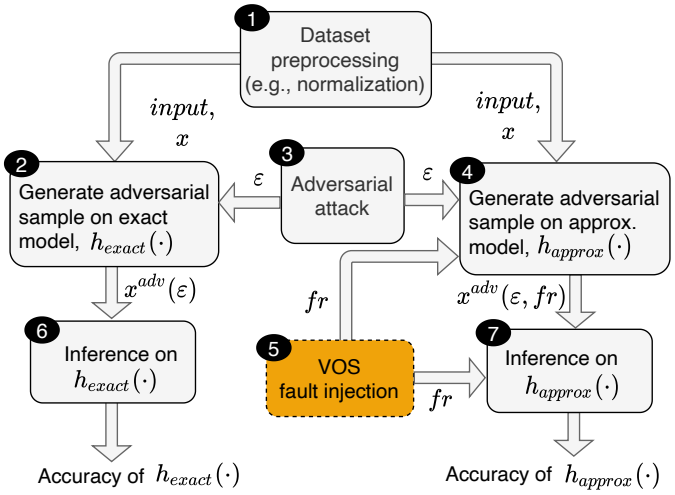


Fig. 2: Proposed methodology flowchart

A. Experimental Setup

For all our experiments, we use the open source machine learning framework PyTorch [32] running on a server, which comprises Intel Broadwell Xeon E5 processor with 16 cores and 512 GB RAM.

Attack configurations: We run the ℓ_2 and ℓ_∞ configuration of C&W, PGD, and HSJ attacks. We implement these attacks using Adversarial Robustness Toolbox (ART) [33] in PyTorch.

Benchmark networks: We consider the following networks in our evaluation: **(a)** LeNet-5 CNN with MNIST dataset [34], which is a collection of $70,000$ gray scale images (60,000 for training and $10,000$ for testing). MNIST comprises 10 classes corresponding to 10 digits. The architecture of LeNet-5 has three convolutions (CONV) layers and two fully connected (FC) layers. **(b)** AlexNet image classification CNN with CIFAR-10 dataset [35], which contains a total of $60,000$ color images representing 10 different classes; for each class, it has 5000 training images and 1000 testing images. AlexNet consists of five CONV layers and three FC layers. **(c)** ResNet-18 with CIFAR-100 [36], which is a deeper CNN architecture used with more challenging dataset compared to the previous two CNNs and datasets. ResNet-18 has 17 CONV layers and one FC layer, and CIFAR-100 is a collection of $60,000$ color images representing 100 classes.

Exact models: After normalizing the dataset, we train the exact model of LeNet-5, AlexNet, and ResNet-18 using a normalized MNIST, CIFAR-10, and CIFAR-100 datasets respectively.

Approximate models: We use the same trained CNNs of the exact models for their respective approximate models. The difference is that approximate models are implemented using controllable stochastic fault injection in the CONV layers. The error rate is accordingly fixed to the VOS level.

Evaluation framework: For robustness evaluation under VOS, we elaborate an exploration framework that we outline in Figure 2. We refer to the exact model and approximate models as $h_{exact}(\cdot)$ and $h_{approx}(\cdot)$, respectively in Figure 2. Once the models are trained, we proceed to generating adversarial attacks, namely C&W, PGD, and HSJ attack, targeting the exact model and approximate model. These attacks produce adversarial examples: x^{adv}. It is to be noted that every VOS level, i.e., fault rate, corresponds to a separate model-under-test since it has its specific behavior. We generated adversarial attacks on approximate models for various fault rates: $fr \in \{10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}\}$. To accurately report the experimental results for the approximate models, since their behaviour is stochastic, we repeat the experiments 10 times and reported the mean and standard deviation. In all attack variants where the ART toolbox provides the parameter ε, we vary ε and measure the classification accuracy of the exact model and approximate models.

B. Resilience against white-box attacks

In this section, we show the security analysis of CNNs under white-box attacks, particularly, using the powerful C&W and PGD attacks with ℓ_∞ and ℓ_2 variants.

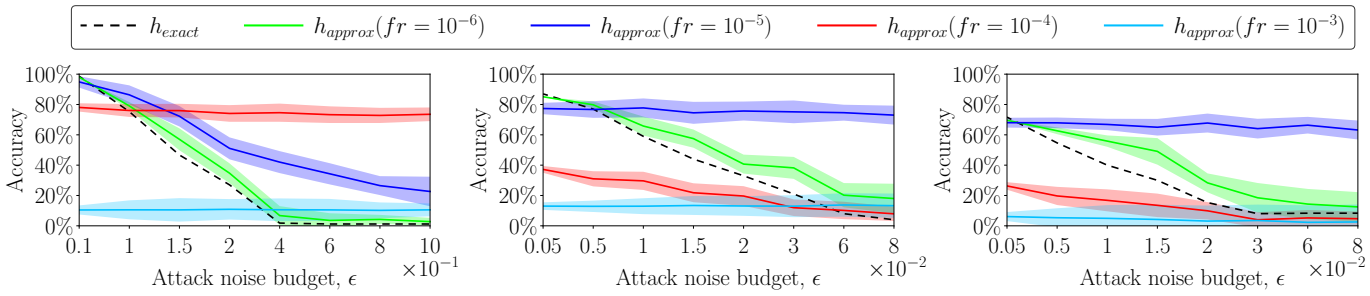
Figure 3, shows the classification accuracy of the exact model and approximate models of the LeNet-5, AlexNet, and ResNet-18 CNNs under ℓ_∞ C&W attack while varying ε. For LeNet-5 (Figure 3-(a)), the exact model (h_{exact}) yields very high classification accuracy (around 99%) when ε is very low (0.01). However, while we increase ε, the exact model accuracy decreases until it is nearly totally fooled after $\varepsilon = 0.4$. Most importantly, approximate model $fr = 10^{-4}$ maintains a high detection accuracy of about 78.06% while varying ε (even when the exact model is totally fooled, $\varepsilon = 0.4$), which provides considerable robustness. Interestingly, this observation holds for AlexNet (Figure 3-(b)) and ResNet-18 (Figure 3-(c)) for approximate model $fr = 10^{-5}$ as they maintain a classification accuracy of 77.75% and 68.01%, respectively.

Figure 4 shows the robustness for ℓ_2 C&W; we cannot vary ε for this attack since the ART toolbox does not provide this parameter. In particular, the ART toolbox generates adversarial samples for a fixed epsilon, which totally fools the exact model of LeNet-5, and the exact model accuracy of AlexNet and ResNet-18 drops to about 10%. Interestingly, approximate model $fr = 10^{-5}$ yields the highest accuracy (with 82.51% for LeNet-5, 81.09% for AlexNet and 66.04% for ResNet-18) across all models for ℓ_2 C&W.

Again, Figure 5 shows the robustness results against ℓ_∞ PGD attack. This is the only attack variant where minimal VOS-induced faults, i.e., $fr = 10^{-6}$, achieves the maximum robustness across all models. Specifically, the approximate model $fr = 10^{-6}$ on LeNet-5, AlexNet, and ResNet-18 shows the robustness of about 94%, 83% and 66%, respectively.

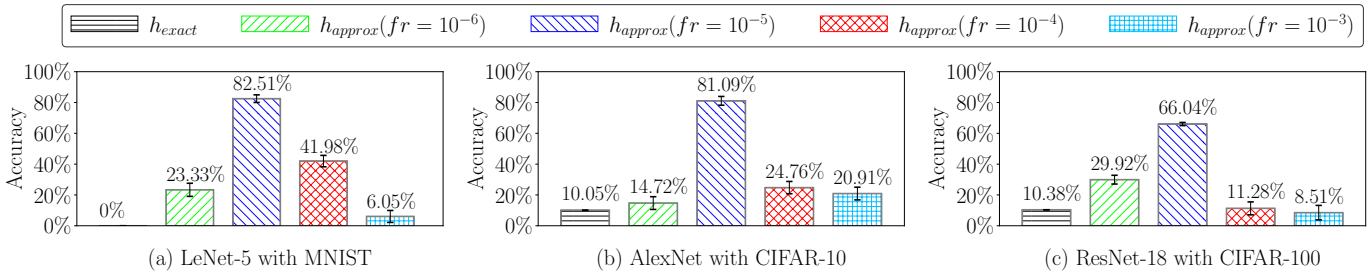
Similarly, Figure 6 shows the robustness against ℓ_2 PGD attack, where approximate model $fr = 10^{-4}$ achieves the highest accuracy of about 88% for LeNet-5. For AlexNet and ResNet-18, approximate model $fr = 10^{-5}$ performs the best with about 81% and 67% accuracy, respectively.

Although white-box setting is the strongest attack scenario since the adversary has access to model gradients, our approach can guarantee significantly high robustness across all



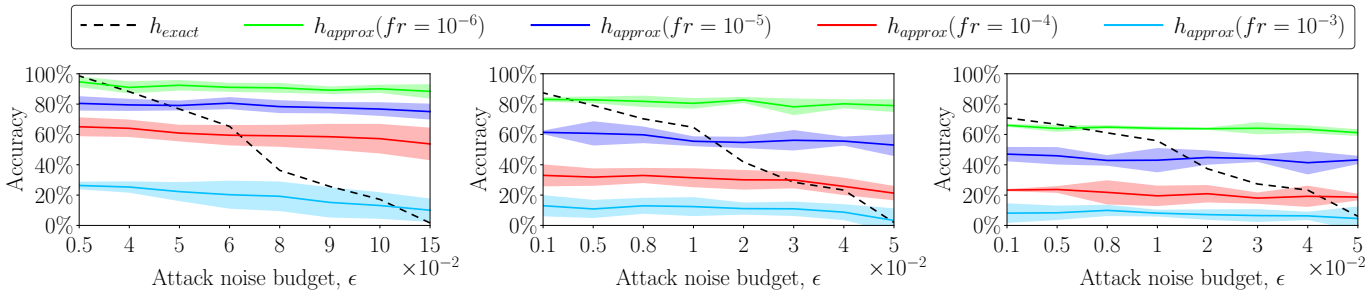
(a) LeNet-5 with MNIST (b) AlexNet with CIFAR-10 (c) ResNet-18 with CIFAR-100

Fig. 3: Robustness against ℓ_∞ C&W attack.



(a) LeNet-5 with MNIST (b) AlexNet with CIFAR-10 (c) ResNet-18 with CIFAR-100

Fig. 4: Robustness against ℓ_2 C&W attack.



(a) LeNet-5 with MNIST (b) AlexNet with CIFAR-10 (c) ResNet-18 with CIFAR-100

Fig. 5: Robustness against ℓ_∞ PGD attack.

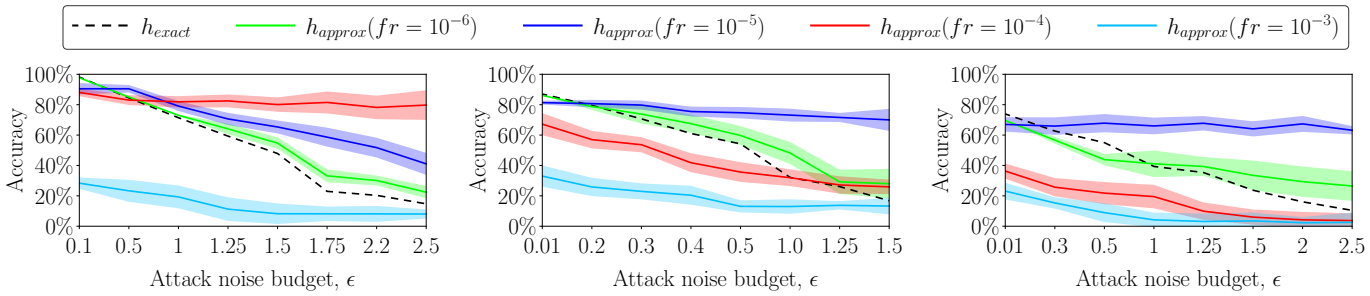
attack variants (from ℓ_2 to ℓ_∞ of C&W and PGD), across all models (from shallow to deep), and across all datasets (from easy to challenging) considered.

C. Resilience against black-box attacks

Unlike white-box attacks, in HSJ black-box attack, the adversary do not have access to the noise budget. Thus, we run the HSJ attack with the default configuration, which uses 40 iterations with ℓ_2 norm and 100 iterations with ℓ_∞ norm. Figures 7 and 8, shows classification accuracy of the exact model and approximate models of the LeNet-5, AlexNet, and ResNet-18 CNNs under ℓ_∞ HSJ and ℓ_2 HSJ, respectively. For the exact model, all results (figures) show that they are almost totally fooled as their detection accuracy is close to zero. For LeNet-5 (Figures 7-(a) and 8-(a)), approximate model $fr = 10^{-4}$ is the most robust, yielding an average of 79.63% and 81.59% accuracy, respectively. For AlexNet (Figure 7-(b) and 8-(b)), approximate model $fr = 10^{-5}$ shows the highest robustness with 77.69% and 79.58% accuracy, respectively. For ResNet-18 (Figures 7-(c) and 8-(c)), approximate model $fr = 10^{-5}$ shows the highest robustness with 65.04% and 66.32% accuracy, respectively.

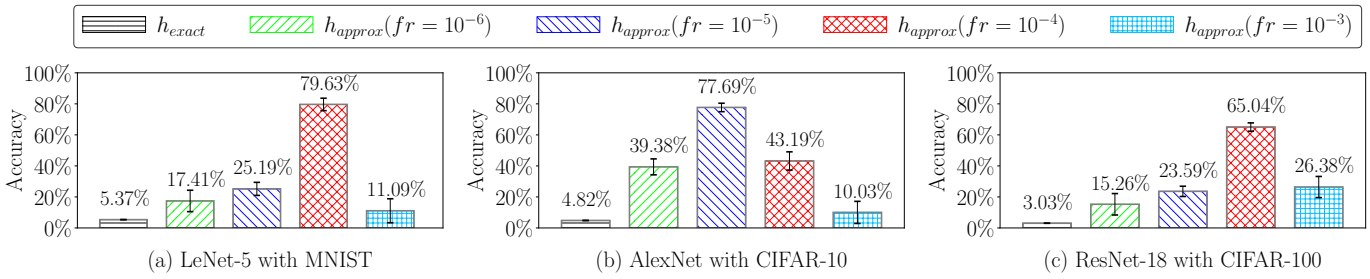
An interesting observation is that AlexNet performs the best at fault rate of 10^{-5} but LeNet-5 at fault rate of 10^{-4}; this is because of the fact that we inject faults after each convolution (CONV) layer, and AlexNet has 5 CONV layers while LeNet-5 has only 3 CONV layers. Out of 5 CONV layers, AlexNet receives substantial stochastic noise even for lower fault rates, while LeNet-5 requires greater fault rates to reach the same saturation out of 3 CONV layers.

Insight: From the comparison of black-box and white-box settings, we notice an interesting counter-intuitive property. In fact, we recorded higher robustness under white-box than black-box setting. This is counter-intuitive since the former is theoretically stronger than the latter, at least from the attacker's knowledge perspective. We believe that this is due



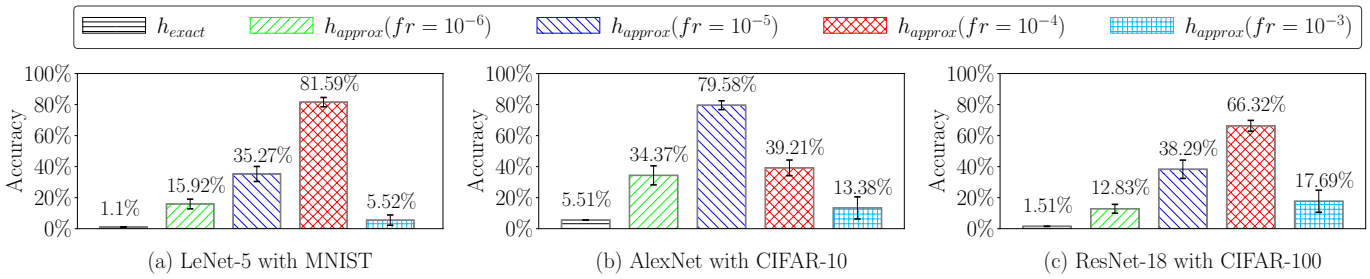
(a) LeNet-5 with MNIST (b) AlexNet with CIFAR-10 (c) ResNet-18 with CIFAR-100

Fig. 6: Robustness against ℓ_2 PGD attack.



(a) LeNet-5 with MNIST (b) AlexNet with CIFAR-10 (c) ResNet-18 with CIFAR-100

Fig. 7: Robustness against ℓ_∞ HSJ attack.



(a) LeNet-5 with MNIST (b) AlexNet with CIFAR-10 (c) ResNet-18 with CIFAR-100

Fig. 8: Robustness against ℓ_2 HSJ attack.

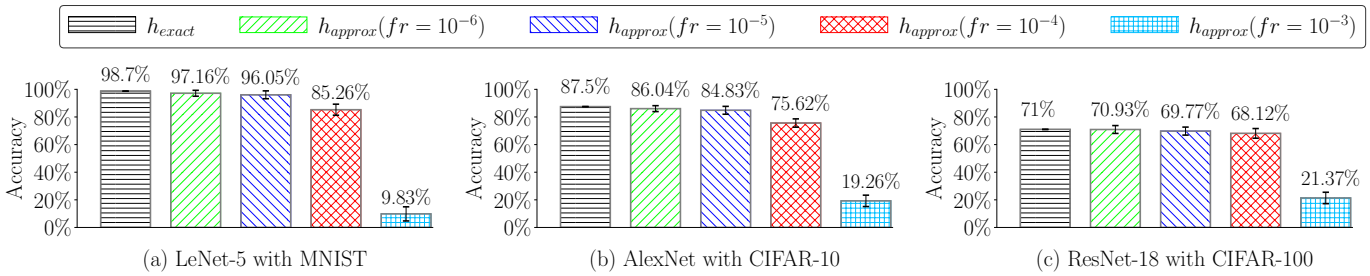
to the impact of VOS on the actual gradient. In fact, the gradient ∇_x, which is the derivative of logits with respect to input, is impacted by the internal computation stochasticity, in every single attack generation iteration. However, the black-box does not have access to the internal feature maps and is, by consequence, less impacted by the internal stochasticity. It is more impacted by the variations in the output, which appear more clearly with the higher error rates.

VII. Accuracy, Security and Power Trade-off

When evaluating a defense for DNNs, it is important to evaluate the baseline accuracy of the protected models (under benign input) besides their security (under adversarial input) since there is no point of using a robust model against adversarial attacks that performs poorly when classifying clean input (which is the case that you expect most of the time). Therefore, after evaluating the security aspect of VOS, in this section, we will study the impact of VOS on the baseline accuracy of the models. In addition, we will evaluate the impact of VOS on power consumption.

Baseline accuracy: Figure 9 presents the model baseline accuracy under different VOS levels, compared to the corresponding exact (unprotected) CNNs. Interestingly, the baseline accuracy of the approximate models are very comparable to the accuracy of the exact model when the computational faults rates are not very aggressive. In contrast, when the computational faults rates are more aggressive (e.g., $fr = 10^{-3}$), the baseline accuracy of the approximate models drops significantly. Nonetheless, by analysing the security results, the faults rates that achieves the highest robustness resulted in 2.65%, 2.67%, and 2.88% accuracy loss for LeNet-5, AlexNet, and ResNet-18, respectively. In this case, the accuracy drop can be seen as the cost for security.

Power savings: An interesting property of using VOS as a defense is that it naturally offers power savings since we are reducing the supply voltage. To demonstrate the power saving advantage, we implement a multiply-accumulate (MAC) circuit at 45 nm technology with PTM [37] using Keysight Advanced Design System (ADS) platform and run Monte Carlo simulations under VOS considering process and thermal



(a) LeNet-5 with MNIST (b) AlexNet with CIFAR-10 (c) ResNet-18 with CIFAR-100

Fig. 9: Baseline accuracy of the tested models under different VOS levels

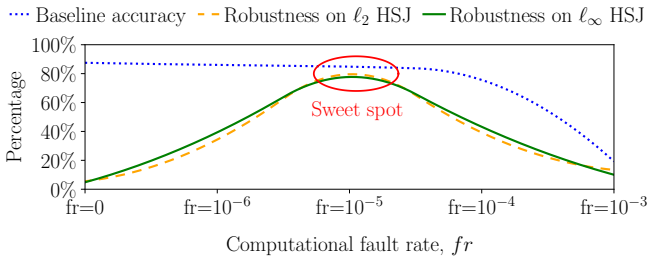


Fig. 10: An illustration of the accuracy/robustness tradeoff for AlexNet with CIFAR-10 on HSJ. In the figure, $fr = 0$ indicates the exact model, h_{exact}.

variation. This allows us to evaluate the computational faults rate as a function of VOS level. Figure 11 shows the dynamic power savings corresponding to the multiplier accuracy loss. The results demonstrated that a considerable power gains come with down-scaling the supply voltage.

Trade-off: Our results show that a tradeoff between accuracy and robustness with by-product power savings could be found, thereby ensuring robust models with low accuracy cost and by-product power savings. An example of possible "sweet-spots" that can be found for a CNNs is given in Figure 10 for Alexnet under HSJ attacks. We could not include all tradeoff results for all networks due to the space limitation. Nonetheless, the example shows that a quick exploration can be done for a given CNN to achieve the highest possible robustness with the lowest possible accuracy drop.

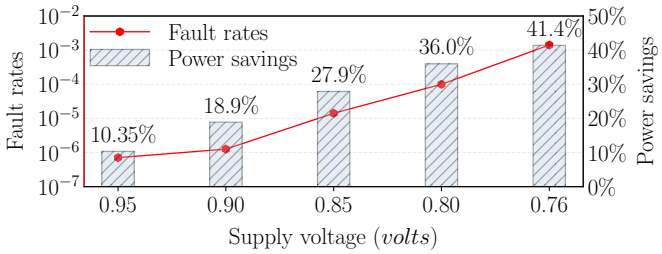


Fig. 11: MAC timing error rates and corresponding power saving under different supply voltage levels.

VIII. Related Work

With the emergence of different adversarial attacks that target DNNs, researchers proposed several defenses that can be grouped into the following categories:

Adversarial Training (AT). Goodfellow et al. [10] proposed to train the model with adversarial samples to harden it against adversarial attacks. Subsequent related methods combined AT with other techniques such as cascade adversarial training [11] and principled training [12]. While AT is efficient on known attacks, it cannot detect new attack strategies, and generating adversarial samples (for training) is computation-intensive and requires additional time for model fitting [38].

Input Preprocessing. Input preprocessing depends on applying transformations to the input to remove the adversarial perturbations [13]–[15], [39]. Nonetheless, these defenses are vulnerable to white-box attack, where the attackers have access to the gradient, including the preprocessing unit. This is because of the fact that, given the knowledge of gradients and transformations, attackers are able to reverse the preprocessing and thus restore malicious perturbations.

Gradient Masking (GM): GM can be traced back to defensive distillation [16] that uses gradient masking and model regularization. This technique has been undermined by Carlini & Wagner attack [21]. In [40], authors proposed to regularize the gradient input by penalizing variations in the model output with respect to changes in the input during the training of differentiable models. Nonetheless, GM requires re-training and gives a false sense of security, as shown in [41].

Randomization-based defenses. These techniques are the closest to our work [17]–[20]. Liu et al. [19] suggest to randomize the entire DNN and predict using an ensemble of multiple copies of the DNN. Lecuyer et al. [17] also suggest to add random noise to the first layer of the DNN and estimate the output by a Monte Carlo simulation. These techniques offer a bounded theoretical guarantee of robustness. Furthermore, a concurrent work showed that VOS helps securing malware detectors against evasion attacks [7], [42]. Finally, Guesmi et al. [43] demonstrated that data-dependent noise injection can help the robustness of DNNs against adversarial attacks.

IX. Conclusion

This work utilizes, for the first time, VOS to enhance DNNs security under adversarial attack settings. The randomness generated by stochastic timing violations is hereby leveraged as a practical defense mechanism that transforms CNNs into moving targets. We believe that this finding advances the state of the art, especially for embedded systems and Edge-dedicated machine learning applications. In fact, in addition to

its security advantages, our method comes with a by-product power saving.

REFERENCES

[1] Mohammed Al-Qizwini, Iman Barjasteh, Hothaifa Al-Qassab, and Hayder Radha. Deep learning algorithm for autonomous driving using googlenet. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 89–96. IEEE, 2017.

[2] Mohammad Badrul Alam Miah, Mohammad Abu Yousuf, Md Sohag Mia, and Md Parag Miya. Handwritten courtesy amount and signature recognition on bank cheque using neural network. *International Journal of Computer Applications*, 118(5), 2015.

[3] Ivan Evtimov, Kevin Eykholt, Earlence Fernandes, Tadayoshi Kohno, Bo Li, Atul Prakash, Amir Rahmati, and Dawn Song. Robust physical-world attacks on machine learning models. *arXiv preprint arXiv:1707.08945*, 2(3):4, 2017.

[4] Valerio Venceslai, Alberto Marchisio, Ihsen Alouani, Maurizio Martina, and Muhammad Shafique. Neuroattack: Undermining spiking neural networks security through externally triggered bit-flips. *arXiv preprint arXiv:2005.08041*, 2020.

[5] Md Shohidul Islam, Khaled N Khasawneh, Nael Abu-Ghazaleh, Dmitry Ponomarev, and Lei Yu. Efficient hardware malware detectors that are resilient to adversarial evasion. *IEEE Transactions on Computers*, 2021.

[6] Md Shohidul Islam, Behnam Omidi, and Khaled N Khasawneh. Monotonic-hmds: Exploiting monotonic features to defend against evasive malware. In *2021 22nd IEEE International Symposium on Quality Electronic Design (ISQED)*, pages 97–102.

[7] Md Shohidul Islam, Ihsen Alouani, and Khaled N Khasawneh. Enhancing hardware malware detectors' security through voltage over-scaling. In *2021 5th ACM SIGARCH Workshop on Cognitive Architectures*.

[8] Md Shohidul Islam, Abraham Peedikayil Kuruvila, Kanad Basu, and Khaled N Khasawneh. Nd-hmds: Non-differentiable hardware malware detectors against evasive transient execution attacks. In *2020 IEEE 38th International Conference on Computer Design (ICCD)*, pages 537–544.

[9] Khaled N Khasawneh, Nael B Abu-Ghazaleh, Dmitry Ponomarev, and Lei Yu. Adversarial evasion-resilient hardware malware detectors. In *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 1–6. IEEE, 2018.

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

[11] Taesik Na, Jong Hwan Ko, and Saibal Mukhopadhyay. Cascade adversarial machine learning regularized with a unified embedding, 2017.

[12] Aman Sinha, Hongseok Namkoong, and John Duchi. Certifying some distributional robustness with principled adversarial training, 2017.

[13] Nilaksh Das, Madhuri Shanbhogue, Shang-Tse Chen, Fred Hohman, Li Chen, Michael E Kounavis, and Duen Horng Chau. Keeping the bad guys out: Protecting and vaccinating deep learning with jpeg compression. *arXiv preprint arXiv:1705.02900*, 2017.

[14] Margarita Osadchy, Julio Hernandez-Castro, Stuart Gibson, Orr Dunkelman, and Daniel Pérez-Cabo. No bot expects the deepcaptcha! introducing immutable adversarial examples, with applications to captcha generation. *IEEE Transactions on Information Forensics and Security*, 12(11):2640–2653, 2017.

[15] Shixiang Gu and Luca Rigazio. Towards deep neural network architectures robust to adversarial examples. *arXiv preprint arXiv:1412.5068*, 2014.

[16] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 582–597. IEEE, 2016.

[17] M. Lecuyer, V. Atlidakis, R. Geambasu, D. Hsu, and S. Jana. Certified robustness to adversarial examples with differential privacy. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 656–672, 2019.

[18] Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. Certified adversarial robustness via randomized smoothing. In *Proceedings of the 36th International Conference on Machine Learning*, pages 1310–1320, 2019.

[19] Xuanqing Liu, Minhao Cheng, Huan Zhang, and Cho-Jui Hsieh. Towards robust neural networks via random self-ensemble, 2017.

[21] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE symposium on security and privacy (sp)*, pages 39–57. IEEE, 2017.

[20] Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. Certified defenses against adversarial examples, 2018.

[22] Xiaoyong Yuan, Pan He, Qile Zhu, and Xiaolin Li. Adversarial examples: Attacks and defenses for deep learning. *IEEE transactions on neural networks and learning systems*, 30(9):2805–2824, 2019.

[23] Martin Wirnshofer. *Variation-Aware Adaptive Voltage Scaling for Digital CMOS Circuits*. Springer, isbn 978-94-007-6196-4, 2013.

[24] Beng Chiew Tang. *Security Engineering of Hardware-Software Interfaces*. PhD thesis, Columbia University, 2018.

[25] RightMark Gathering. Rmclock utility., 2019. Accessed online July 2020 at http://cpu.rightmark.org/products/rmclock.shtml.

[26] Miha Eleršič. Guide to linux undervolting for haswell and never intel cpus, 2019. Accessed online July 2020 at https://github.com/mihic/linux-intel-undervolt.

[27] Ali Dasdan and Ivan Hom. Handling inverted temperature dependence in static timing analysis. *ACM Trans. Des. Autom. Electron. Syst.*, 11(2):306–324, April 2006.

[28] R. Li, R. Naous, H. Fariborzi, and K. N. Salama. Approximate computing with stochastic transistors' voltage over-scaling. *IEEE Access*, 7:6373–6385, 2019.

[29] Jianbo Chen, Michael I Jordan, and Martin J Wainwright. Hopskipjumpattack: A query-efficient decision-based attack. In *2020 IEEE symposium on security and privacy (sp)*, pages 1277–1294. IEEE, 2020.

[30] 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.

[31] Naveed Akhtar and Ajmal Mian. Threat of adversarial attacks on deep learning in computer vision: A survey. *Ieee Access*, 6:14410–14430, 2018.

[32] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems*, pages 8026–8037, 2019.

[33] Maria-Irina Nicolae, Mathieu Sinn, Minh Ngoc Tran, Beat Buesser, Ambrish Rawat, Martin Wistuba, Valentina Zantedeschi, Nathalie Baracaldo, Bryant Chen, Heiko Ludwig, et al. Adversarial robustness toolbox v1. 0.0. *arXiv preprint arXiv:1807.01069*, 2018.

[34] Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. 2010. *URL http://yann. lecun. com/exdb/mnist*, 7:23, 2010.

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

[36] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 and cifar-100 datasets. *URl: https://www. cs. toronto. edu/kriz/cifar. html*, 6:1, 2009.

[37] Nanoscale Integration and Modeling (NIMO) Group. Predictive technology model (ptm) website, January 2012. Accessed online November 2019 at http://ptm.asu.edu.

[38] Khaled N Khasawneh, Nael Abu-Ghazaleh, Dmitry Ponomarev, and Lei Yu. Rhmd: Evasion-resilient hardware malware detectors. In *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 315–327, 2017.

[39] Amira Guesmi, Ihsen Alouani, Mouna Baklouti, Tarek Frikha, and Mohamed Abid. SIT: Stochastic Input Transformation to defend against adversarial attacks on deep neural networks. *IEEE Design Test*, pages 1–1, 2021.

[40] Andrew Slavin Ross and Finale Doshi-Velez. Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients. In *Thirty-second AAAI conference on artificial intelligence*, 2018.

[41] Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018*, July 2018.

[42] Md Shohidul Islam, Ihsen Alouani, and Khaled N Khasawneh. Stochastic-hmds: Adversarial resilient hardware malware detectors through voltage over-scaling. *arXiv preprint arXiv:2103.06936*, 2021.

[43] Amira Guesmi, Ihsen Alouani, Khaled N Khasawneh, Mouna Baklouti, Tarek Frikha, Mohamed Abid, and Nael Abu-Ghazaleh. Defensive approximation: securing cnns using approximate computing. In *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 990–1003, 2021.