SecureVolt: Enhancing Deep Neural Networks Security via Undervolting

Md Shohidul Islam, *Student Member, IEEE,* Ihsen Alouani, *Member, IEEE,* and Khaled N. Khasawneh, *Member, IEEE*

***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 over-scaling (VOS) as a lightweight and efficient defense against adversarial attacks. Specifically, we exploit the stochastic timing violations of VOS within computing elements 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. We propose a space exploration to identify a possible trade-off between robustness, accuracy and power gains. Furthermore, we observe the behavior of models' epistemic uncertainty under variable undervolting aggressiveness. Our experiments show that model uncertainty analysis is coherent with the observation in our robustness/accuracy exploration.**

***Index Terms*—Voltage overscaling, adversarial attack, DNN security, moving-target defense, power savings**

I. Introduction

PRESENT day tech-dominated world is witnessing a significantly widespread deployment of Deep Learning in diverse sectors. Deep Neural Networks (DNNs) and especially Convolutional Neural Networks (CNNs) deliver promising performance in several challenging applications. However, they are found vulnerable to adversarial attacks, *i.e.*, models output wrong prediction for adversarial input samples, which are maliciously crafted by injecting low-magnitude noise that is often imperceptible to human eyes. The consequence of such misprediction can be dramatic. For example, in autonomous vehicles, mistakenly classifying *stop* sign as *yield* or *speed limit* sign can implies potential human or material damage. In fact, attackers exploited this vulnerability and demonstrated practical attacks in multiple domains: self-driving cars [1] detecting *stop* sign as 45 MPH speed limit sign; e-diagnosis system [2] detecting benign medical image as malignant and vice versa; hardware malware detector [3]–[7] detecting malware as regular software; face detection system [8] confused by impersonation, voice-controlled system [9] misunderstanding speech commands, *e.g.*, detecting "*how are you*" as "*open the door*"; copyright detection [10] failing to discern protected and non-protected contents, etc. Thus, the issue has raised serious concern especially for safety-critical areas and security-sensitive applications.

Md Shohidul Islam is with the ECE Department at George Mason University, Fairfax, VA, USA and with the CSE Department at Dhaka University of Engineering & Technology, Gazipur, Bangladesh. Khaled N Khasawneh is with the ECE Department, George Mason University, Fairfax, VA, USA. E-mail:{mislam20, kkhasawn}@gmu.edu

Ihsen Alouani is with the Centre for Secure Information Technologies (CSIT) in Queen's University Belfast, UK. E-mail: i.alouani@qub.ac.uk

Researchers and practitioners have proposed various defense solutions, to harden CNNs against adversarial attacks, that largely fall into the following four groups: Adversarial Training (AT) [11], Input Preprocessing [12]–[14], Gradient Masking (GM) [15], and Randomization-based Defenses [16]–[18]. In AT, adversarial samples were used to train target models so that models can identify attacks during inference time. Although known attacks can be detected by AT, new/unknown attack variants cannot be detected by this defense technique. Moreover, AT requires huge time in generating the attack samples (for using in the victim training) and demands extra time for model fitting, rendering the solution time- or compute-intensive and thus challenging. 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 [19]. Randomization based defenses introduce random noise in the entire DNNs or in some select layers [16], 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. [18] 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 [16], [17], 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, but with by-product gain in power savings, our method shows promising results under strong white-box and black-box attack settings.
4) We also evaluate the effect of undervolting-induced stochastic fault injection on prediction from a different perspective. Specifically, we measure the epistemic uncertainty, *i.e.,* prediction uncertainty caused by the randomized model, which is found low at the optimal fault rate for accuracy/robustness tradeoff.
5) For research reproducibility and to encourage the community to further explore this technique, we plan to open-source our code.

The remainder of the paper is organized as follows. Section II briefly presents the necessary background; Section III illustrates the VOS-induces computational faults characterization on real CPU; Section IV discusses the threat model considered; Section V describes the proposed methodology; Section VI thoroughly analyzes the security evaluation of our defense; Section VII devises a sample way of finding trade-off between accuracy, robustness, and power consumption; Section VIII explores the epistemic uncertainty of model prediction under voltage overscaling; Section IX summarizes the related work; and finally Section XI concludes the paper.

II. Background

A. Adversarial Attacks

Adversarial attack samples are generated by slightly perturbing the inputs in order to confuse the target classifier. For instance in computer vision domain, adversarial examples are crafted by injecting small noise, which often are not perceptible to human eyes, into the original/clean image; such adversarial inputs are capable of fooling the victim classifier completely, resulting in incorrect predicted labels. Putting it formally, suppose $h(\cdot)$ is an $m-ary$ CNN classifier used for image classification, x is an original/clean image, c is the ground truth label of x such that $c = h(x)$, where $c \in \{1, 2, ..., m\}$. Next, the attacker's target is to inject small perturbation to x that will generate adversarial image, x', so that the predicted label for x' is different from c. With this information, the adversarial sample is generated by solving the below optimization problem [20].

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

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, $p = 0$ indicates ℓ_0-norm, which counts the number of pixels with different values at corresponding locations; $p = 2$ indicates ℓ_2-norm, which measures the Euclidean distance and $p = \infty$ indicates $\ell_\infty-$norm, which is the Chebyshev distance measuring the maximum difference for all pixels at corresponding locations. Suppose $\{x_1, x_2, ..., x_N\}$ is the original image set and $\{x'_1, x'_2, ..., x'_N\}$ is the corresponding adversarial image set, then the attack success rate is measured using the following equation:

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

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 [21]:

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

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 [21]:

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

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

This section is dedicated to explore VOS-induced fault characterization 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.

We experimented using an Intel Broadwell processor (model number i7-5557U) running on Ubuntu 16.04 LTS with stock Linux v4.15. Luckily, the reverse engineering effort [22], [23] revealed that the operating voltage of modern processor cores can be controlled from software through undocumented model specific register (MSR). As such, we used the MSR to control the voltage of the CPU from software. 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 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.

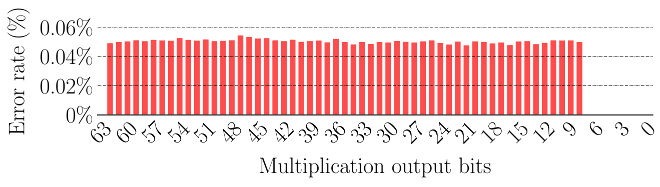

Fig. 1: Probability distribution of faulty bits location for undervolted multiplication results. These results are generated on i7-5557U at 2.2 GHz, with a CPU temperature of 49° C and the CPU was undervolted by $-130\ mV$.

A. VOS effect on multiplication

To understand how multiplication operation is effected by VOS, we performed an experiment where we executed a multiplication several times without changing the operand values.

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	0x042d704da4ae35e8	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

The experiment is repeated for different sets of operands, and some selected results (*i.e.*, for faulty multiplications) are reported in Table I. Each group (i.e., 3 rows) of results shown in Table I was observed at a certain voltage. The first group showed faults for 0.873 Volt, the second group for 0.861 Volt, and the third for 0.847 Volt. Specifically, for each multiplication, we undervolted by a step of 1 mV starting from the nominal supply voltage until we notice the first computing errors. Then we maintain the voltage fixed and re-execute the same multiplications and observe the computing errors before resuming the VOS. We observed faults in the multiplication result while the voltage was reduced by -103 mV to -145 mV. We observed faults in multiplication result while the voltage was reduced by -103 mV to -145 mV as compared to the base voltage. We noticed that the location of faults varied randomly for the same operands. More interestingly, we found that the pair of operands generating faults in one run sometimes generate correct results in another run. As such, these confirm that the VOS induces stochastic faults.

To examine the distribution of fault locations induced by undervolting, we conducted an experiment using the above mentioned experiment and calculated the fault rates for each bit. This allowed us to gain insight into the distribution of errors at a particular level of undervolting. The findings are presented in Figure 1 (details of the experimental setup provided in the caption). From this experiment, we observed that the result sign bit was not flipped at all; it is because of the fact that the multiplication result sign bit is just a simple XOR operation of operands' sign bits, which is far from the critical path of the multiplier circuit. Likewise, we did not observe any fault in the 8 least significant bits of the multiplication result, mainly because the propagation delay is small in such case.

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 [24]. 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 2 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.

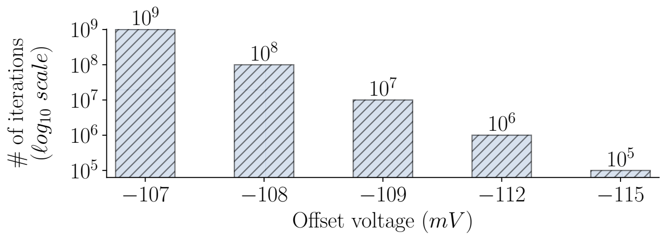

Fig. 2: Number of multiplications required to observe a single computational fault vs. necessary voltage offset from $nominal - V_{dd}$ on i7-5557U at 2.2 GHz

B. VOS effect on other operations

Similar to multiplication operations, we performed the same investigation on other operations such as *addition, subtraction, bit-wise AND, OR,* and *Shift* operations. However, these operations did not generate any fault while reducing the supply voltage. This is because of the fact that these operations require simpler circuit design, which have smaller propagation delay in comparison with multiplication instruction.

IV. Threat Model

This section describes the threat model under which we evaluate the DNN models' robustness. For this experiment, two threat settings have been considered including the Black-box attack and White-box attack scenarios explained as follows.

Black-box attack.

Under this threat model, attacker can query the target models and observe their prediction for given inputs. In other words, attacker can access the target model's input and output but does not have access to the victim model's internal information, *e.g.*, model architecture, gradient, or hyperparameters. We implement the latest and the most powerful attack called *HopSkipJump (HSJ)* [25], as a representative of the black-box attack family. HSJ is an iterative attack that utilizes only the decision of target model rather than model gradient information. By relying on the victim's final output, HSJ estimates model gradients and approximates the direction of gradient by using the binary information at the decision boundary.

White-box attack.

White-box threat model assumes a more powerful attacker than the black-box. Under this settings, attacker can access input/output, target model's architecture, gradients, weights. bias, etc. Adversarial attacks are systematically generated by utilizing the victim model's internal information. Because of the unrestricted access to model information, white-box attacks are more difficult to defend against. Under this threat model, we implemented Carlini & Wagner ($C\&W$) attack [19] and Projected gradient descent (PGD) attack [26], because of their superior strength.

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

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

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 [11], 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. These attacks have been summarized in Table II.

TABLE II: Summary of the used attack methods. Notice that the strength estimation (out of 5 stars) is based on [27].

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

We propose to harden DNN models against adversarial attacks by leveraging hardware-induced noise, specifically through VOS. We outline below the rationale behind choosing VOS.

(i) It injects stochastic behavior: Injecting random noise's positive impact on DNNs robustness has been proven theoretically in [16], [17]. 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 [16], [17], 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 10. 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 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 the multiplication operations. The impact of VOS-induced faults is simulated by fault injection at the output of multiply operations with different error rates to assess the impact of different voltage levels. We modified the core convolution functions in PyTorch to support this approach.

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

In this section, we begin with the explanation of different setups of our experiment (*cf.* Section VI-A). Later, we thoroughly illustrate the effectiveness of our defense, which shows how much resilient are the target models against attack. Specifically, we evaluate the robustness of defending models against white-box attack (*cf.* Section VI-B) and against black-box attack (*cf.* Section VI-C) in the presence of our defense.

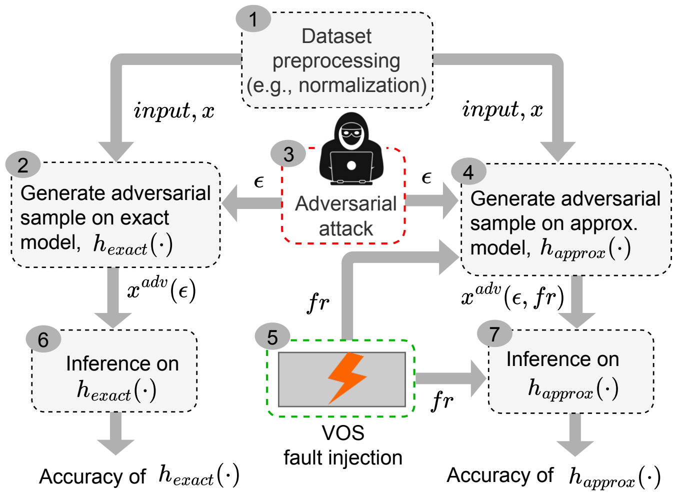
1
Dataset preprocessing (e.g., normalization)
$input, x$
$input, x$
2
Generate adversarial sample on exact model, $h_{exact}(\cdot)$
3
Adversarial attack
ϵ
ϵ
4
Generate adversarial sample on approx. model, $h_{approx}(\cdot)$
$x^{adv}(\epsilon)$
fr
$x^{adv}(\epsilon, fr)$
6
Inference on $h_{exact}(\cdot)$
5
VOS fault injection
fr
7
Inference on $h_{approx}(\cdot)$
Accuracy of $h_{exact}(\cdot)$
Accuracy of $h_{approx}(\cdot)$

Fig. 3: Different steps of the proposed exploration framework. Here, ϵ is the adversarial attacker noise budget and fr is the VOS-induced fault rate.

A. Experimental Setup

This section describes how we setup the experiment and step-by-step flow of the execution. Our experiments aim to generate adversarial attacks on victim DNNs, comparatively with and without our proposed defense, and report robustness. We conducted our entire exploration using PyTorch, running on a server, which comprises Intel Broadwell Xeon E5 processor with 16 cores and 512 GB RAM.
Attack configurations: We implement both the ℓ_2 and ℓ_∞ variant of C&W, PGD, and HSJ attacks. We choose these attacks because of their strong attack performance, as shown in Table II. We perform these attacks using PyTorch-based tool called Adversarial Robustness Toolbox (ART) [28].
Benchmark networks and datasets: We explored our defense on five image classifiers using four different datasets briefly summarized as follows. **(a)** LeNet-5 + MNIST: Here the model has a total of 5 layers comprising 3 convolution layers and 2 fully connected layers. The model is trained and tested on a 10-class dataset (MNIST), which has 70,000 28×28 grayscale images of handwritten single digit between "0" and "'9'. More specifically, it has 60,000 training images and 10,000 test images, where each class has 6,000 training and 1,000 test samples. **(b)** AlexNet + CIFAR-10: Here the model has a total of 8 layers divided into 5 convolution layers and 3 fully connected layers. The model is trained and tested on a 10-class dataset (CIFAR-10), which has 60,000 32×32 color images with 6000 images per class. More specifically, it has 50,000 training samples and 10,000 test samples, where each class has 5,000 training and 1,000 test samples. **(c)** ResNet-18 + CIFAR-100: Here the model has a total of 18 layers split into 17 convolution layers and 1 fully connected layer. The model is trained and tested on a 100-class dataset (CIFAR-100) comprising 60,000 32×32 color images with 600 images per class. More specifically, it has 50,000 training samples and 10,000 test samples, where each class has 500 training and 100 test samples. **(d)** ResNet-50 + ImageNet: Here the model

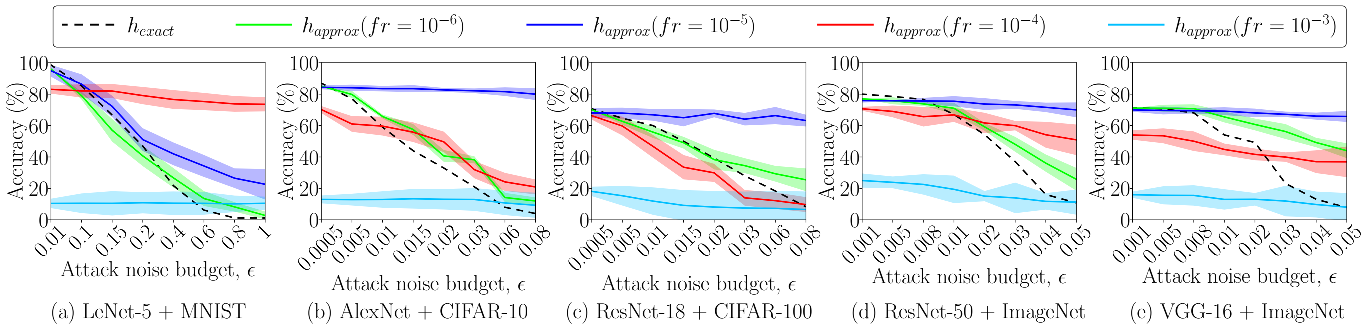
Fig. 4: Robustness against ℓ_∞ C&W attack.

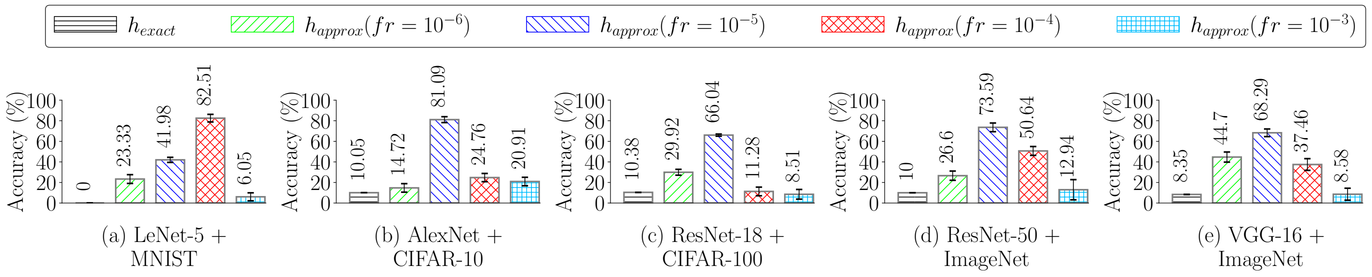
Fig. 5: Robustness against ℓ_2 C&W attack.

has a total of 50 layers grouped into 49 convolution layers and 1 fully connected layer. The model is trained and tested on a 1000-class dataset (ImageNet), which is a benchmark in image classification and object detection; and finally **(e)** VGG-16 (having 13 CONV layers and 3 FC layers) with ImageNet dataset. The reason behind choosing different networks is that they represent shallow, deep, and deeper network. Again, we choose MNIST, CIFAR-10, and CIFAR-100 because they are representative of easier to challenging datasets. Our goal is to explore whether our defense is effective across diverse attack variants (ℓ_2 and ℓ_∞), across different threat models (white-box and black-box), across different models (from shallow to deeper), and across various datasets (from easy to challenging).

Exact models: We trained the exact (*i.e.*, conventional) model of LeNet-5 on MNIST, AlexNet on CIFAR-10, ResNet-18 on CIFAR-100, ResNet-50 on ImageNet, and VGG-16 on ImageNet. We normalized the datasets before training the models.

Approximate models: We implement the approximate models by injecting controllable stochastic faults after CONV layers of the exact models. The learnable parameters (*i.e.*, weight, bias, etc.) of approximate models are updated with the values of that of their corresponding trained exact models.

Evaluation framework: Figure 3 outlines different stages of our experimental framework to evaluate model robustness under VOS. In the figure, the notation $h_{exact}(\cdot)$ indicates the exact model and $h_{approx}(\cdot)$ represents the approximate model. We start with dataset preprocessing (*e.g.*, normalization) and model training. Next, we run adversarial attacks (*e.g.*, C&W, PGD, and HSJ attack) on the exact model and approximate model to generate adversarial samples. Adversarial attacks perturb clean/original inputs by injecting noise, with a noise budget of ϵ. For attacking exact model, we vary only the ϵ; however, for attacking approximate model, we vary both ϵ and fault rate fr. As such, $x^{adv}(\epsilon)$ is the adversarial sample on exact model and $x^{adv}(\epsilon, fr)$ is the adversarial sample on approximate model. We implement approximate models for different fault rates, $fr \in \{10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}\}$, corresponding to different VOS levels. It is to be mentioned that every fault rate or VOS level generates a specific behavior in the model, indicating a separate model-under-test. After generating adversarial samples, we measure the detection accuracy on exact model (*cf.* Figure 3-Step 6) and approximate model (*cf.* Figure 3-Step 7). Because the approximate models show stochastic behavior, in order to get representative results, we repeat each experiment on approximate model 10 times and report the mean and standard deviation of accuracy. Standard deviation reflects the amount of stochasticity.

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 4, shows the classification accuracy of the exact model and approximate models of the LeNet-5, AlexNet, ResNet-18, ResNet-50, and VGG-16 CNNs under ℓ_∞ C&W attack while varying ϵ. For LeNet-5 (Figure 4-(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 $\epsilon = 0.4$. 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, $\epsilon = 0.4$), which provides considerable robustness.

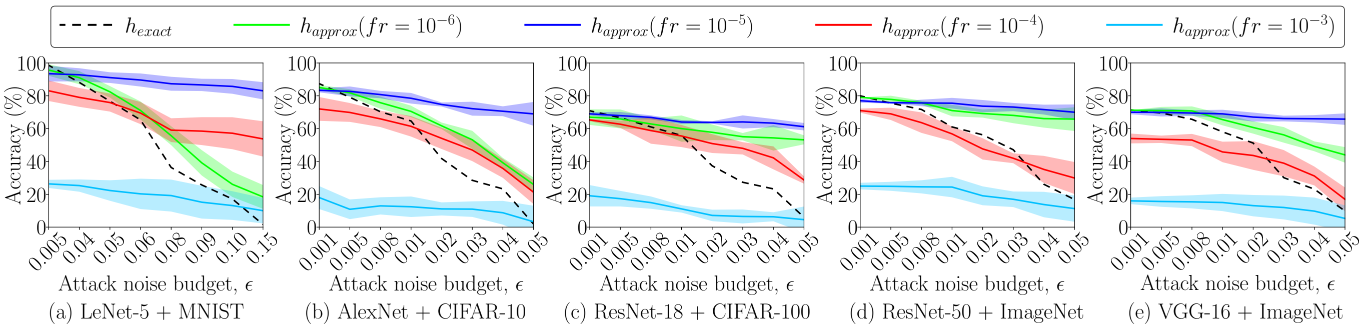

Fig. 6: Robustness against ℓ_∞ PGD attack.

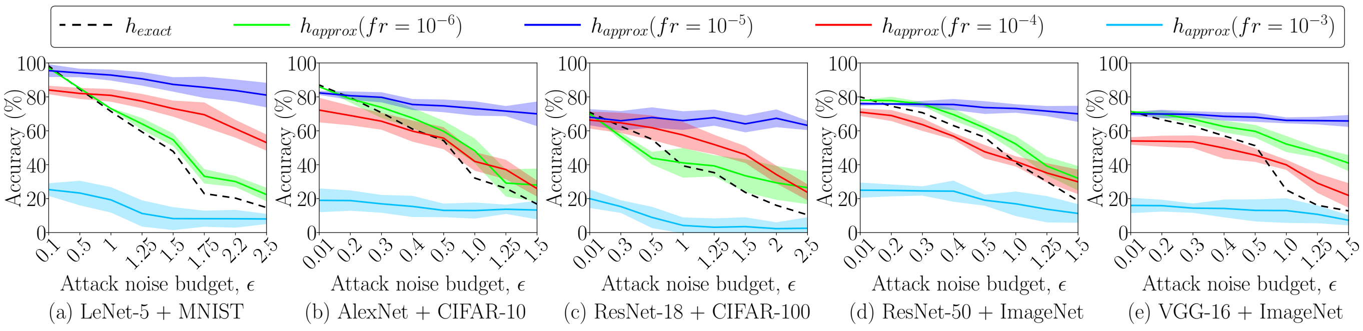

Fig. 7: Robustness against ℓ_2 PGD attack.

Interestingly, this observation holds for other networks for $fr = 10^{-5}$, which helps them maintain a classification accuracy of ∼82.70%, ∼67.77%, ∼73.70%, and ∼68.05%, in Figure 4-(b), Figure 4-(c), Figure 4-(d), and Figure 4-(e) respectively.

Figure 5 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 ResNet-18, and ResNet-50 drops to about ∼10%, and that of VGG-16 drops to ∼8%. Interestingly, approximate model fr=10^{-4} yields the highest robustness accuracy of ∼82.51% for LeNet-5 and fr=10^{-5} makes others models robust— with ∼81.09% robustness for AlexNet in Figure 5-(b), ∼66.04% for ResNet-18 in Figure 5-(c), ∼73.59% for ResNet-50 in Figure 5-(d), and ∼68.29% for VGG-16 in Figure 5-(e).

Again, Figure 6 shows the robustness results against ℓ_∞ PGD attack. This is the only attack variant where fr=10^{-5} yields the maximum robustness across all models. Specifically, the approximate model fr=10^{-5} on LeNet-5, AlexNet, ResNet-18, ResNet-50, and VGG-16 shows the robustness accuracy of ∼93%, ∼83%, ∼69%, ∼77%, and ∼70%, respectively.

Similarly, Figure 7 shows the robustness against ℓ_2 PGD attack, where approximate model fr=10^{-5} maintains the highest robustness accuracy across all models considered. Specifically, against this attack variant, LeNet-5 shows the robustness of ∼93% in Figure 7-(a), AlexNet shows ∼82% in Figure 7-(b), ResNet-18 shows ∼68% in Figure 7-(c), ResNet-50 shows ∼76% in Figure 7-(d), and VGG-16 shows ∼70% in Figure 7-(e).

Despite white-box attack being the strongest setting due to the availability of model internal information, our defense still enables the protected models to maintain considerably high accuracy on adversarial samples. In fact, we observe the high robustness for all the considered models and against all variants of C&W and PGD attacks. The exact model accuracy in all figures does not have any standard deviation since the exact model does not include our stochastic fault injection and thus shows deterministic behavior.

C. Resilience against black-box attacks

Unlike white-box attacks, the adversary do not have access to the noise budget in HSJ black-box attack. Thus, we run the HSJ attack with the default configuration, which uses 40 iterations with ℓ_2 norm and 100 iterations with ℓ_∞ norm. For all models, ℓ_∞ HSJ attack (i.e., Figures 8) used $\epsilon = 0.27$ and ℓ_2 HSJ attack (i.e., Figure 9) used $\epsilon = 3.5$. We updated Section. Figures 8 and 9 shows the classification accuracy of exact model and approximate models of all CNNs considered 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 8-(a) and 9-(a)), approximate model fr=10^{-4} is the most robust, yielding an average of ∼79.63% and ∼81.59% accuracy, respectively. Similarly, for AlexNet (Figure 8-(b) and 9-(b)), approximate model fr=10^{-5} shows the highest robustness with ∼77.69% and ∼79.58% accuracy, respectively. Likewise, for ResNet-18 (Figures 8-(c) and 9-(c)), approximate model fr=10^{-4} shows the highest robustness

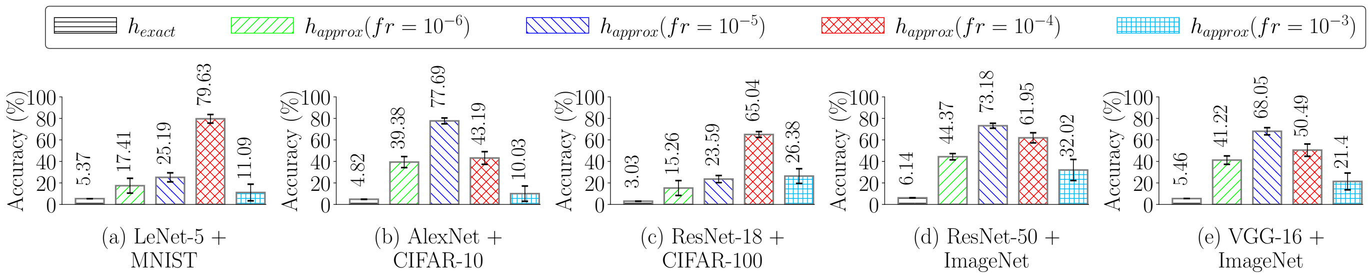

Fig. 8: Robustness against ℓ_∞ HSJ attack.

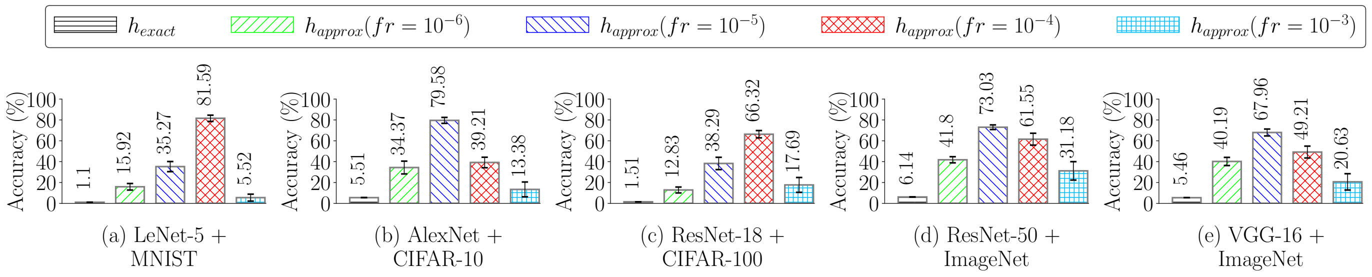

Fig. 9: Robustness against ℓ_2 HSJ attack.

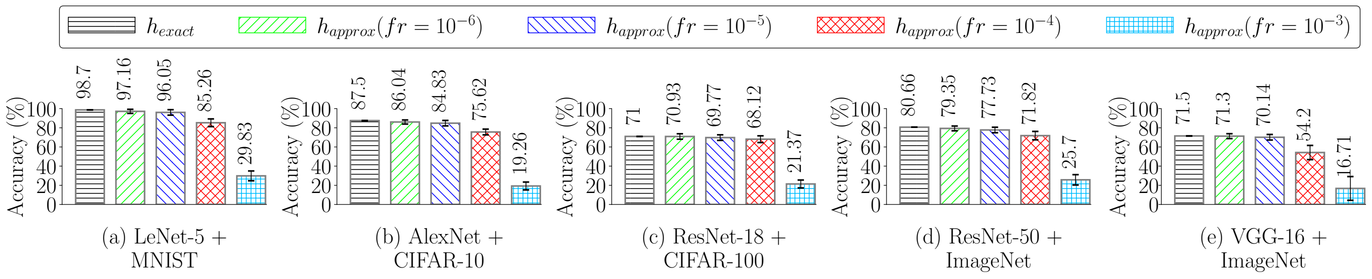

Fig. 10: Baseline accuracy of the tested models under different VOS levels (*i.e.* fault rates)

with ∼65.04% and ∼66.32% robustness accuracy, respectively. Moreover, for ResNet-50 (Figures 8-(d) and 9-(d)), approximate model fr=10^{-5} shows the highest robustness with ∼73.18% and ∼73.03% robustness accuracy, respectively. Furthermore, for VGG-16 (Figures 8-(e) and 9-(e)), approximate model fr=10^{-5} shows the highest robustness with ∼68.05% and ∼67.96% robustness accuracy, respectively

An interesting observation for HSJ attack is that LeNet-5 performs the best at fault rate of 10^{-4} but AlexNet, ResNet-50, and VGG-16 yield the highest robustness at fault rate of 10^{-5}; this is because of the fact that we inject faults after each convolution (CONV) layer, and LeNet-5 has only 3 CONV layers while others have more. Comparatively deeper models receive substantial stochastic noise even for lower fault rates, while LeNet-5 requires greater fault rates to reach the same saturation out of only 3 CONV layers. The same observations holds between ResNet-18 and ResNet-50, where the former is comparatively shallow model than the latter.

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 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.

D. *Transferability of Attacks*

So far, we considered generating adversarial samples based on approximate models, specifically targeting a particular fault rate, and then evaluated on the same model. However, here we evaluated the transferability of attacks, where adversarial samples are generated on one model (the exact model) and tested on another model (the approximate model). The heatmap presented in Figure 11 illustrates the results in terms of adversarial accuracy (expressed as a percentage). Notably, the heatmap reveals the presence of a ”sweet spot” in fault rates that ensures a high level of adversarial accuracy is maintained.

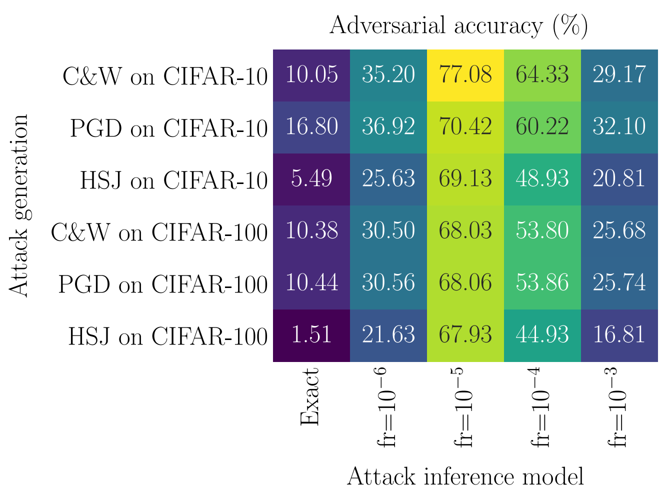

Fig. 11: Transferability of attacks generated on exact model. Results in the heatmap represent the adversarial accuracy (in percentage). Heatmaps show the results for ℓ_2 variants of C&W ($\epsilon = 1.35$), PGD ($\epsilon = 2.5$), and HSJ ($\epsilon = 3.5$) attacks on CIFAR-10 and CIFAR-100 datasets.

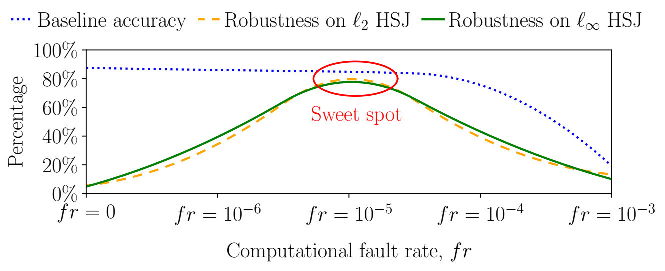

Fig. 12: An illustration of the accuracy/robustness tradeoff for AlexNet with CIFAR-10 on HSJ. In the figure, $fr = 0$ indicates the exact model, h_{exact}.

VII. Accuracy, Security and Power Trade-off

For any defense solution, it is necessary to investigate how the defense is affecting secured model's baseline accuracy (on clean inputs) besides its robustness (on malicious inputs). Because it is of no use deploying a robust model that has low classification accuracy on benign inputs. It is especially important since models are more frequently expected to classify benign inputs than malicious inputs. Thus, after assessing security/robustness aspect of VOS (*cf.* Section VI-B and Section VI-C, we delve into the effect of VOS on model baseline accuracy in this section. Furthermore, we analyze the effect of VOS on power consumption.

Baseline accuracy: Baseline accuracy refers to the classification accuracy on clean/original inputs; in other words, it is the accuracy without attack scenario. Figure 10 shows the baseline accuracy for various VOS levels, contrasting with the respective exact (undefended) models. We observe that nominal fault rates result in approximate models yielding comparable baseline accuracy like the exact model. However, more aggressive fault rates (*e.g.*, fr=10^{-3}) cause significant drop in accuracy of the approximate models. Nevertheless, from the security analysis, the faults rates that achieves the highest robustness resulted in ∼2.65%, ∼2.67%, ∼2.88%, ∼2.93%, and ∼1.36% accuracy loss for LeNet-5, AlexNet, ResNet-18, ResNet-50, and VGG-16, respectively. We interpret such loss in accuracy as the cost for security.

Power savings: Since the supply voltage is reduced in VOS, using such technique as a defense also intrinsically offers by-product savings of energy/power consumption. To demonstrate the power saving advantage, we implement a multiply-accumulate (MAC) circuit at 45 nm technology with PTM [29] using Keysight Advanced Design System (ADS) platform and run Monte Carlo simulations under VOS considering process and thermal variation. This allows us to evaluate the computational faults rate as a function of VOS level. Figure 13 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 CNN is given in Figure 12 for Alexnet under HSJ attacks. We show Figure 12 as a sample that can be followed to obtain all tradeoff results for all networks; however, we could not include them all to avoid repetition. 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.

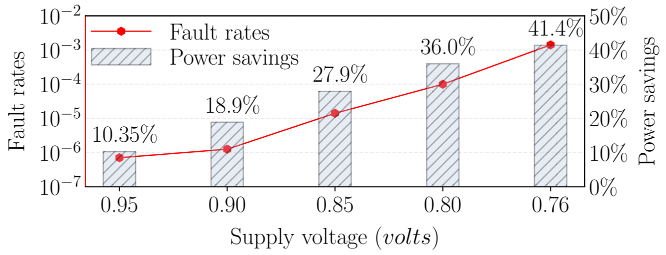

Fig. 13: MAC timing error rates and corresponding power saving under different supply voltage levels.

VIII. Epistemic uncertainty

This section explores the uncertainty of model prediction under voltage overscaling. It is important to analyze uncertainty because models can be uncertain even if they assign a high probability to a predicted class [30]. In practice, uncertainty may arise due to the input or model. In our defense approach, we inject stochastic faults into the inference computation of the model rather than the input. Thus, we estimate the model uncertainty or epistemic uncertainty, which represents the uncertainty when predicting a single data point with a randomized model. We measure the epistemic uncertainty by following the approach illustrated in [30]. Given a DNN model $h : \mathcal{X} \rightarrow \mathcal{Y}$, an input image $x \in \mathcal{X}$, and a predicted class $y \in \mathcal{Y}$, the epistemic uncertainty is defined by the softmax variance. For this, we do N_m stochastic forward passes for each image x and obtain N_m softmax vectors. Then the model uncertainty (MU) is estimated by the following equation.

$$MU(x) = \frac{1}{|\mathcal{Y}|}\sum_{i=1}^{|\mathcal{Y}|}\frac{1}{N_m}\sum_{j=1}^{N_m}(p_{ij}-\bar{p}_i)^2 \tag{7}$$

Where p_{ij} is the probability of predicting the input x as the i-th class in the j-th forward pass and $\bar{p}_i$ is the mean probability for the i-th class over N_m passes, *i.e.*, $\bar{p}_i = \frac{1}{N_m}\sum_{j=1}^{N_m} p_{ij}$. We used N_m=100 for all models in our experiment.

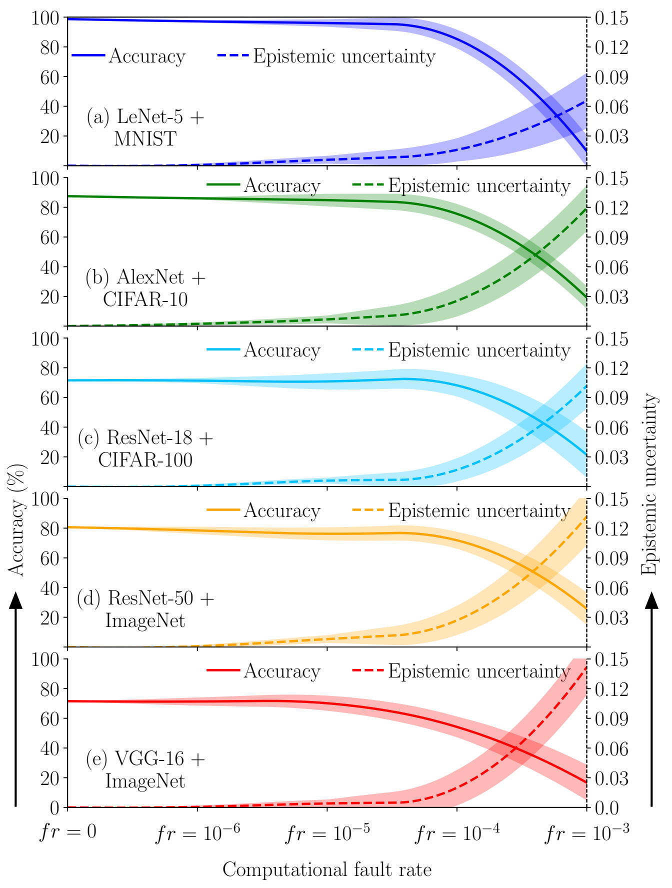

Fig. 14: Effect of fault rates on *accuracy* and *epistemic uncertainty* of different models— (a) LeNet-5, (b) AlexNet, (c) ResNet-18, (d) ResNet-50, and (e) VGG-16. For each sub-figure, left side Y-axis indicates the *accuracy* and right side Y-axis indicates the *epistemic uncertainty*; the X-axis (*i.e.,* fault rate) is shared among all sub-figures.

We measure the epistemic uncertainty for each image in *test dataset* while injecting stochastic faults in the model; for each fault rate, we calculated the mean and standard deviation of the uncertainty. Figure 14 shows the epistemic uncertainty on the right-side Y-axis and contrasts it with model accuracy on the left-side Y-axis. Figure 14 comprises five sub-figures, one sub-figure per evaluated model. All sub-figures share a common X-axis representing different fault rates. More specifically, Figure 14(a) is for LeNet5+MNIST, Figure 14(b) is for AlexNet+CIFAR-10, Figure 14(c) is for ResNet-18+CIFAR-100, Figure 14(d) is for ResNet-50+ImageNet, and Figure 14(e) is for VGG-16+ImageNet.

For each sub-figure, the solid line indicates mean accuracy, the broken line indicates mean epistemic uncertainty, and the shadow around the line indicates corresponding standard deviation. In each sub-figure, fr=0 refers to the exact model, where we did not inject any stochastic faults in the inference computation, resulting in *zero* standard deviation for both the accuracy and uncertainty. As we start injecting faults with increasing fault rates, we observe increases in the standard deviation. Likewise, the mean epistemic uncertainty increases with the increasing faults, making the models more uncertain about their prediction, resulting in decreasing trend in their detection accuracy. Interestingly, the fault rate that is optimal for accuracy/robustness tradeoff corresponds to low uncertainty and high accuracy. For instance, we devise a sample tradeoff between accuracy and robustness for AlexNet classifier on CIFAR-10 dataset in Figure 12, which shows that fault rate fr=10^{-5} turns out to be optimal. The impact of this optimal fault rate fr=10^{-5} on epistemic uncertainty for this model is shown in Figure 14(b), which shows an epistemic uncertainty of 0.0068. Besides, at this fault rate, the model maintains a classification accuracy of 84.43%, which is ∼2.67% loss in accuracy compared to the baseline accuracy. We observe similar trend for other models as can be seen in Figure 14.

IX. Related work

This section presents the related work on different defenses methods.

Adversarial Training (AT). Adversarial training was proposed by Goodfellow et al. [11] and Madry et al. [31], where adversarial samples were used to train target models so that models can identify attacks during inference time. Although known attacks can be detected by AT, new/unknown attack variants cannot be detected by this defense technique. Moreover, AT requires huge time in generating the attack samples (for using in the victim training) and demands extra time for model fitting [32].

Input Preprocessing. Das et al. [12], Osadchy et al. [13], Gu et al. [14], and Guesmi et al. [33] proposed input preprocessing based defenses, where adversarial perturbations are removed from the input by means of applying various transformations. However, while black-box attacks are defeated by this defense, whit-box attacks are still effective, where the knowledge about gradients and preprocessed inputs are available to adversaries. It happens because the insight of gradients and transformation enables the adversaries to restore adversarial perturbations by reversing the defensive preprocessing.

Gradient Masking (GM): Under this category of defense, Papernot et al. [15] introduced defensive distillation, where gradient masking and regularization of model has been used. However, later, Carlini & Wagner published the powerful C&W attack [19], which severely undermines the defensive distillation defense. Furthermore, Ross and Doshi-Velez [34] proposed another defense along this line; specifically, authors penalize model output variation with respect to input changes, in order to perform regularization of gradient input; authors perform these steps for training models that are differentiable. Nevertheless, Athalye et al. [35] exposed the downside of GM as it depends on model re-training and shows a false sense of security.

Hardware-based defenses: Hardware-based defenses were proposed to overcome the performance overhead introduced by software-based defenses. 2-in-1 accelerator [36] showed that adversarial attacks transfer poorly between different precisions of an adversarially trained model. Thus, they built a ML accelerator that can randomly switch between adversarially trained model during inference. Defensive Approximation [37], proposed to inject input dependent noise using approximate multipliers that is able to make the model robust against adversarial attacks. However, both works [36], [37] require hardware changes (i.e., building new hardware). Majumdar et al. [38] showed that undervolting an FPGA introduce computational noise to the model inference that makes the model more robust against adversarial attacks. Compered to our work, this work focused on FPGAs while our work focused on CPUs. In addition, we provide full characterization of the faults.

Randomization-based defenses. Lecuyer et al. [16], Cohen et al. [17] and Raghunathan et al. [18] porposed randomization based defenses, which are the closest to our proposed defense. Specifically, the entire CNN has been randomized and an ensemble of several copies of the CNNs were used for inference. Likewise, the first layer of CNN is randomized by injecting random noise in [16] and a Monte Carlo simulation is used for output estimation. A bounded formal guarantee of robustness is achieved by those techniques. Moreover, Islam et al. [4], [5] demonstrated that malware detectors can be secured via VOS against evasive malware attacks. Furthermore, defensive approximation [37] is another defense, proposed by Guesmi et al., which empirically showed that CNNs can be harden against adverasarial samples by injecting noise that are data dependent.

TABLE III: Comparison of defenses

Defenses	retraining free?	inference overhead free?	specialized hardware independent?	results in power savings?	low effect on baseline accuracy?
Adversarial Training	✗	✗	✓	✗	✗
Input Preprocessing	✗	✗	✓	✗	✗
Gradient Masking	✗	✗	✓	✗	✗
Hardware based defense	✓	✓	✗	✓	✗
Randomization based defense	✗	✗	✓	✗	✗
Our defense	✓	✓	✓	✓	✓

X. Discussion

This work has unprecedentedly shown that a promising security enhancement technique of CNN/DNN can be implemented in a practical and efficient manner using undervolting.

Calibration: The faults induced by undervolting can differ between devices, making it necessary to perform a distinct calibration for each one to identify the optimal level of undervolting that balances accuracy and robustness. Additionally, temperature plays a crucial role in influencing these faults, as suggested by the findings in [39]. As a result, the undervolting level should be adaptive based on the temperature to maintain the optimal tradeoff between accuracy and robustness.

Limitations: Models that perform computations on values that are close to zero cannot be protected because the least significant bits cannot be flipped via undervolating.

Effect of stochastic faults on the variance of the reported results: We also investigate how the mean and standard deviation of accuracy are effected if we increase the number of iterations of our experiments. Therefore, repeated the experiments while increasing the number of iterations to 150 rather than 10. To understand the effect, we subtracted the mean of 10 iterations and from that of 150 iterations. We found some changes in mean accuracy for the approximate models due to their stochasticity. Our results shows that the fluctuation in accuracy of the approximate models are negligible (i.e., less than 1%).

XI. Conclusion

We propose an effective defense technique, by unprecedentedly leveraging VOS, to enhance the security of CNN/DNN against adversarial samples. VOS generates randomness in DNN computation by violating the timing requirement deliberately and stochastically, which transforms the DNNs into moving target defense. The proposed defense does not require re-training or additional computation and can be used with pre-trained models. Moreover, besides the security advantage, the proposed defense offers a by-product energy savings due to its intrinsic undervolting and cutting power consumption, As such, based on the empirical results, we believe that our defense advances the sate of the art, especially for the machine learning applications targeting edge/embedded/IoT devices.

Acknowledgment

The work in this paper is partially supported by National Science Foundation grant CCF-2212427, and by EdgeAI KDT-JU European project (101097300).

References

[1] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno, and D. Song, "Robust physical-world attacks on deep learning visual classification," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 1625–1634.

[2] S. G. Finlayson, J. D. Bowers, J. Ito, J. L. Zittrain, A. L. Beam, and I. S. Kohane, "Adversarial attacks on medical machine learning," *Science*, vol. 363, no. 6433, pp. 1287–1289, 2019.

[3] M. S. Islam, B. Omidi, and K. N. Khasawneh, "Monotonic-hmds: Exploiting monotonic features to defend against evasive malware," in *2021 22nd IEEE International Symposium on Quality Electronic Design (ISQED)*, pp. 97–102.

[4] M. S. Islam, I. Alouani, and K. N. Khasawneh, "Enhancing hardware malware detectors' security through voltage over-scaling," in *2021 5th ACM SIGARCH Workshop on Cognitive Architectures*.

[5] ——, "Stochastic-hmds: Adversarial-resilient hardware malware detectors via undervolting," in *60th Design Automation Conference (DAC)*, 2023.
[6] M. S. Islam, A. P. Kuruvila, K. Basu, and K. N. Khasawneh, "Nd-hmds: Non-differentiable hardware malware detectors against evasive transient execution attacks," in *2020 IEEE 38th International Conference on Computer Design (ICCD)*, pp. 537–544.
[7] K. N. Khasawneh, N. B. Abu-Ghazaleh, D. Ponomarev, and L. Yu, "Adversarial evasion-resilient hardware malware detectors," in *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2018, pp. 1–6.
[8] M. Sharif, S. Bhagavatula, L. Bauer, and M. K. Reiter, "Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition," in *Proceedings of the 2016 acm sigsac conference on computer and communications security*, 2016, pp. 1528–1540.
[9] Y. Gong and C. Poellabauer, "Protecting voice controlled systems using sound source identification based on acoustic cues," in *2018 27th International Conference on Computer Communication and Networks (ICCCN)*. IEEE, 2018, pp. 1–9.
[10] P. Saadatpanah, A. Shafahi, and T. Goldstein, "Adversarial attacks on copyright detection systems," in *Proceedings of the 37th International Conference on Machine Learning (ICML 2020)*. PMLR, 2020, pp. 8307–8315.
[11] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.
[12] N. Das, M. Shanbhogue, S.-T. Chen, F. Hohman, L. Chen, M. E. Kounavis, and D. H. Chau, "Keeping the bad guys out: Protecting and vaccinating deep learning with jpeg compression," *arXiv preprint arXiv:1705.02900*, 2017.
[13] M. Osadchy, J. Hernandez-Castro, S. Gibson, O. Dunkelman, and D. Pérez-Cabo, "No bot expects the deepcaptcha! introducing immutable adversarial examples, with applications to captcha generation," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 11, pp. 2640–2653, 2017.
[14] S. Gu and L. Rigazio, "Towards deep neural network architectures robust to adversarial examples," *arXiv preprint arXiv:1412.5068*, 2014.
[15] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *2016 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2016, pp. 582–597.
[16] 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)*, 2019, pp. 656–672.
[17] J. Cohen, E. Rosenfeld, and Z. Kolter, "Certified adversarial robustness via randomized smoothing," in *Proceedings of the 36th International Conference on Machine Learning*, 2019, pp. 1310–1320.
[18] A. Raghunathan, J. Steinhardt, and P. Liang, "Certified defenses against adversarial examples," 2018.
[19] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *2017 IEEE symposium on security and privacy (sp)*. IEEE, 2017, pp. 39–57.
[20] X. Yuan, P. He, Q. Zhu, and X. Li, "Adversarial examples: Attacks and defenses for deep learning," *IEEE transactions on neural networks and learning systems*, vol. 30, no. 9, pp. 2805–2824, 2019.
[21] M. Wirnshofer, *Variation-Aware Adaptive Voltage Scaling for Digital CMOS Circuits*. Springer, isbn 978-94-007-6196-4, 2013.
[22] B. C. Tang, "Security engineering of hardware-software interfaces," Ph.D. dissertation, Columbia University, 2018.
[23] R. Gathering, "Rmclock utility." 2019, accessed online July 2020 at http://cpu.rightmark.org/products/rmclock.shtml.
[24] R. Li, R. Naous, H. Fariborzi, and K. N. Salama, "Approximate computing with stochastic transistors' voltage over-scaling," *IEEE Access*, vol. 7, pp. 6373–6385, 2019.
[25] J. Chen, M. I. Jordan, and M. J. Wainwright, "Hopskipjumpattack: A query-efficient decision-based attack," in *2020 IEEE symposium on security and privacy (sp)*. IEEE, 2020, pp. 1277–1294.
[26] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *arXiv preprint arXiv:1706.06083*, 2017.
[27] N. Akhtar and A. Mian, "Threat of adversarial attacks on deep learning in computer vision: A survey," *Ieee Access*, vol. 6, pp. 14 410–14 430, 2018.
[28] M.-I. Nicolae, M. Sinn, M. N. Tran, B. Buesser, A. Rawat, M. Wistuba, V. Zantedeschi, N. Baracaldo, B. Chen, H. Ludwig *et al.*, "Adversarial robustness toolbox v1. 0.0," *arXiv preprint arXiv:1807.01069*, 2018.
[29] N. Integration and M. N. Group, "Predictive technology model (ptm) website," Jan. 2012, accessed online November 2019 at http://ptm.asu.edu.
[30] F. Yin, Y. Li, C.-J. Hsieh, and K.-W. Chang, "Addmu: Detection of far-boundary adversarial examples with data and model uncertainty estimation," *arXiv preprint arXiv:2210.12396*, 2022.
[31] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *International Conference on Learning Representations (ICLR)*, 2018.
[32] K. N. Khasawneh, N. Abu-Ghazaleh, D. Ponomarev, and L. Yu, "Rhmd: Evasion-resilient hardware malware detectors," in *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*, 2017, pp. 315–327.
[33] A. Guesmi, I. Alouani, M. Baklouti, T. Frikha, and M. Abid, "SIT: Stochastic Input Transformation to defend against adversarial attacks on deep neural networks," *IEEE Design Test*, pp. 1–1, 2021.
[34] A. S. Ross and F. 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.
[35] A. Athalye, N. Carlini, and D. 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*, Jul. 2018. [Online]. Available: https://arxiv.org/abs/1802.00420
[36] Y. Fu, Y. Zhao, Q. Yu, C. Li, and Y. Lin, "2-in-1 accelerator: Enabling random precision switch for winning both adversarial robustness and efficiency," in *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*, 2021, pp. 225–237.
[37] A. Guesmi, I. Alouani, K. N. Khasawneh, M. Baklouti, T. Frikha, M. Abid, and N. 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 (ASPLOS)*, 2021, pp. 990–1003.
[38] S. Majumdar, M. H. Samavatian, K. Barber, and R. Teodorescu, "Using undervolting as an on-device defense against adversarial machine learning attacks," *arXiv preprint arXiv:2107.09804*, 2021.
[39] I. Filanovsky and A. Allam, "Mutual compensation of mobility and threshold voltage temperature effects with applications in cmos circuits," *IEEE Transactions on Circuits and Systems I*, 2001.

Md Shohidul Islam is a Ph.D. student in the Electrical and Computer Engineering Department at George Mason University. He received his Master in Computer Engineering from the University of Ulsan and a Bachelor in Computer Science and Engineering from Rajshahi University of Engineering and Technology. His research interests are in the areas of systems security and adversarial machine learning.

Ihsen Alouani received his Ph.D. in 2016 from Université Polytechnique Hauts-de-France. He is currently a Senior Lecturer at the Centre for Secure Information Technologies (CSIT) in Queen's University Belfast, UK. His research interest includes machine learning security and privacy, neuromorphic computing, and systems reliability and security.

Khaled N. Khasawneh is an Assistant Professor in the Electrical and Computer Engineering Department at George Mason University. His research interests are in architecture support for security and adversarial machine learning. He received his Ph.D. in Computer Science from the University of California at Riverside in 2019.