Defensive Approximation: Enhancing CNNs Security through Approximate Computing

Amira Guesmi
University of Sfax, Tunisia

Ihsen Alouani
Polytechnic University Hauts-De-France, France

Khaled Khasawneh
George Mason University, USA

Mouna Baklouti
University of Sfax, Tunisia

Tarek Frikha
University of Sfax, Tunisia

Mohamed Abid
University of Sfax, Tunisia

Nael Abu-Ghazaleh
University of California Riverside, USA

***Abstract*—In the past few years, an increasing number of machine-learning and deep learning structures, such as Convolutional Neural Networks (CNNs), have been applied to solving a wide range of real-life problems. However, these architectures are vulnerable to adversarial attacks: inputs crafted carefully to force the system output to a wrong label. Since machine-learning is being deployed in safety-critical and security-sensitive domains, such attacks may have catastrophic security and safety consequences. In this paper, we propose for the first time to use hardware-supported approximate computing to improve the robustness of machine learning classifiers. We show that our approximate computing implementation achieves robustness across a wide range of attack scenarios. Specifically, for black-box and grey-box attack scenarios, we show that successful adversarial attacks against the exact classifier have poor transferability to the approximate implementation. Surprisingly, the robustness advantages also apply to white-box attacks where the attacker has access to the internal implementation of the approximate classifier: in this case, we show that substantially higher levels of adversarial noise are needed to produce adversarial examples. We explain some of the possible reasons for this robustness through analysis of the internal operation of the approximate implementation. Furthermore, our approximate computing model maintains the same level in terms of classification accuracy, does not require retraining, and reduces resource utilization and energy consumption of the CNN. We conducted extensive experiments on a set of strong adversarial attacks; We empirically show that the proposed implementation increases the robustness of a LeNet-5 and an Alexnet CNNs by up to 99% and 87%, respectively for strong grey-box adversarial attacks along with up to 67% saving in energy consumption due to the simpler nature of the approximate logic. We also show that a white-box attack requires a remarkably higher noise budget to fool the approximate classifier, causing an average of 4db degradation of the PSNR of the input image relative to the images that succeed in fooling the exact classifier.**

I. Introduction

Convolutional neural networks (CNNs) and other deep learning structures are used in an increasing number of applications owing to their state-of-the-art performance in multiple fields, such as computer vision [1], [2], natural language processing (NLP) [3], robotics [4], autonomous driving [5], and healthcare [6]. While rapid progress in all aspects of CNN development and deployment is occurring, they are vulnerable to adversarial attacks: maliciously designed imperceptible perturbations injected within the input data that cause CNNs to misclassify the data. Adversarial attacks have been demonstrated in real-world scenarios [7], [8], [9], making this vulnerability a serious threat to safety-critical and other applications that rely on CNNs.

Since the demonstration of adversarial examples, several attacks and defenses have been proposed. Previous work proposed solutions based on manipulating either the input data, the objective function, or the network structure to mitigate adversarial effects. However, these proposed techniques require substantial changes to the architecture, retraining procedure, or incorporate additional input data processing overhead. In addition, some proposed defenses have been shown to be vulnerable to alternative attack strategies. We review the state of the art attacks and defenses in Section IX. Another challenge in deploying such systems is their high power and computational requirements, which make them challenging to deploy in embedded systems and latency critical applications. For this reason, many research studies have explored techniques to accelerate CNNs ranging from algorithmic optimizations, hardware acceleration, and even dedicated mobile platforms [10]. Since many of the proposed defenses against adversarial attacks require substantial additional overheads, they make it more challenging to build secure CNNs under power and computational resource constraints.

In this paper, we propose a new hardware based approach to improving the robustness of machine learning classifiers. Specifically, we propose to leverage Approximate Computing (AC), a family of hardware techniques designed to improve the performance and power consumption of logic circuits and processing elements, at the cost of introducing some approximation to their operation. Since CNNs are highly tolerant to computational errors [11], [12], they represent an ideal candidate for approximate implementation. While AC has been explored as a possible approach to reduce power consumption and computation complexity, in this paper, we show for the first time that it can also provide a defense mechanism against adversarial attacks. Our technique, which we call *defensive approximation* (DA), substantially enhances the robustness of CNNs to adversarial attacks. Note that, throughout the paper, we refer to classifiers that uses DA as approximate classifiers. We show that for a variety of attack scenarios, and utilizing

a range of algorithms for generating adversarial attacks, DA provides substantial robustness even under the assumptions of a powerful attacker (e.g., with full access to the classifier structure). Importantly, DA does not require retraining or fine-tuning, allowing pre-trained models to benefit from its robustness and performance advantages by simply replacing the exact multiplier implementations with approximate ones. The approximate classifier achieve similar accuracy to the exact classifier for Lenet-5 and Alexnet. In addition to these attractive properties from a robustness perspective, DA benefits from the conventional advantages of AC, resulting in a less complex design that is both faster and more energy efficient.

We evaluate DA's robustness properties empirically against several threat models and using a range of adversarial example generation algorithms. First, we consider whether an attacker that has access to the exact classifier and generates adversarial examples that fool that classifier, would be able to use those examples against the approximate classifier. We find that these attacks exhibit poor transferability to the approximate classifier (e.g., over 80% of Lenet-5 adversarial examples are classified correctly by the approximate classifier). We then consider two scenarios where the attacker directly attacks the approximate classifier:

- Black-box attack: the attacker reverse engineers the approximate classifier and constructs a proxy of it that uses exact multipliers. Adversarial examples are generated using this proxy model. While these examples transfer back to fool the exact classifier, they are not able to fool the approximate classifier.
- White-box attack: the attacker has full access to the approximate classifier, and can use it to generate examples that reliably fool the approximate classifier. In this case, we show that the amount of injected noise needed to fool the approximate classifier is substantially higher than the noise needed to fool an exact classifier, for example resulting in around 4db degradation of the adversarial example (and 6x increase in Mean Square Error) for DA relative to the ones that fool the exact classifier.

We carry out a number of experiments to better understand the robustness advantages of DA. We show that the unpredictable variations introduced by AC improve the CNN resilience to adversarial perturbations. Experimental results show that DA has a confidence enhancement impact. In fact, the AC-induced noise in the convolution layer is shown to be higher in absolute value when the input matrix is highly correlated to the convolution filter, and by consequence highlights further the features. This observation at the feature map propagates through the model and results in enhanced classification confidence, i.e., the difference between the 1^{st} class and the "runner-up". Intuitively and as shown by prior work [13], enhancing the confidence furthers the classifier robustness.At the same time, we observe negligible accuracy loss compared to a conventional CNN implementation on non-adversarial inputs while providing considerable power savings.

Related to our work, a set of defenses based on randomization emerged over the last years, and provide theoretically quantifiable robustness bounds [13], [14], [15]. Unlike our implementation, these works assume introduction of Gaussian noise at different stages of the operation of the classifier; our approximate multiplier is sensitive to the input data, but is not Gaussian. However, these works support our general observation that the injection of perturbations improves the robustness of the classifier. From a practical perspective, none of these works have been evaluated at scale or with realistic implementations. For example, Raghunathan et al. [15] evaluate only a tiny neural network. Other works [13], [14] consider scalability but require high overhead to implement the defense. Specifically, to estimate the model output, these methods require running a heavy Monte Carlo simulation involving a number of different runs of the CNN for the same input. Our approach is different since, not only our AC-injected noise does not require overhead but comes naturally from the simpler and faster AC implementation. Moreover, while these techniques require additional training, our implementation is a drop-in replacement of the hardware without specific training requirements, and with no changes to the architecture nor the parameters of the CNN.

In summary, the contributions of the paper are:

- We build an aggressively approximate floating point multiplier that injects data-dependent noise within the convolution calculation. Based on this approximate multiplier, we implement an approximate CNN hardware accelerator (Section IV-B).
- To the best of our knowledge, we are the first to leverage AC to enhance CNN robustness to adversarial attacks without the need for re-training, fine-tuning nor input pre-processing. We investigate the capacity of AC to help defending against adversarial attacks in Section V-C.
- We empirically show that the proposed approximate implementation reduces the success rate of adversarial attacks by an average of 87% and 71.5% in Lenet-5 and Alexnet CNNs respectively.
- We illustrate empirically that white-box attacks require substantially higher adversarial perturbations to fool the approximate classifier.
- We provide some insights into the impact of DA through a theoretical analysis (Section VI-B).
- DA is highly practical; it can be deployed without retraining or fine-tuning, achieving comparable classification performance to exact classfiers. In addition to security advantages, DA *improves* performance by reducing latency by 4x, energy by 2.5x, and consequently, the energy delay product by 10x, making it an attractive choice even in Edge device settings (Section VII-C).

II. Background

This section first presents an overview of CNNs, which are the targets of our attack in this paper; however, we expect DA to apply to other learning structures. We then overview adversarial attacks and introduce approximate computing.

A. Convolutional neural networks

A convolutional neural network (CNN) [16] is a Deep Learning algorithm that takes an image as input, automatically extracts features, and produces labels (corresponding to the classification task the CNN was trained for) as output. A CNN consists of convolution layers and sub-sampling layers performing the feature extraction, followed by fully connected layers performing the classification. The input image is fed to the CNN and the convolution layer creates feature maps. The feature maps are pooled (down-sampled) to reduce their dimension while conserving the most important information, which is one of the factors that make CNNs more tolerant to distortions. The fully connected layers use the feature matrix to classify the input. Another important concept of CNNs is the use of activation functions, commonly a rectified linear unit (ReLU). Applying a non-saturating ReLU activation function $ReLu(x) = \max(0, x)$ removes negative values from an activation map by setting them to zero. This increases the nonlinear properties of the decision function and of the overall network without affecting the receptive fields of the convolution layer.

B. Adversarial attacks

Deep learning techniques gained popularity in recent years and are now deployed even in safety-critical tasks, such as recognizing road signs for autonomous vehicles [17]. Despite their effectiveness and popularity, CNN-powered applications are facing a critical challenge adversarial attacks. Many studies [18], [19], [20], [9] have shown that CNNs are vulnerable to carefully crafted inputs designed to fool them, very small imperceptible perturbations added to the data can completely change the output of the model. In computer vision domain, these adversarial examples are intentionally generated images that look almost exactly the same as the original images, but can mislead the classifier to provide wrong prediction outputs. Other work [21] claimed that adversarial examples are not a practical threat to machine learning in real-life scenarios. However, physical adversarial attacks have recently been shown to be effective against CNN based applications in real-world [22].

Minimizing injected noise: Its essential for the adversary to minimize the added noise to avoid detection. For illustration purposes, consider a CNN used for image classification. More formally, given an original input image x and a target classification model $f()$, the problem of generating an adversarial example x^* can be formulated as a constrained optimization [23]:

$$\begin{aligned} x^* = \arg\min_{x^*} \quad & \mathcal{D}(x, x^*), \\ s.t. \quad & f(x) = l,\ f(x^*) = l^*,\ l \neq l^* \end{aligned} \tag{1}$$

Where $\mathcal{D}$ is the distance metric used to quantify the similarity between two images and the goal of the optimization is to minimize this added noise, typically to avoid detection of the adversarial perturbations. l and l^* are the two labels of x and x^*, respectively: x^* is considered as an adversarial example if and only if the label of the two images are different ($f(x) \neq f(x^*)$) and the added noise is bounded ($\mathcal{D}(x, x^*) < \epsilon$ where $\epsilon \geqslant 0$).

Distance Metrics: The adversarial examples and the added perturbations should be visually imperceptible by humans. And since it's hard to model human perception, researchers proposed three metrics to approximate humans perception of visual difference, namely L_0, L_2, and L_∞ [24]. These metrics are special cases of the L_p norm:

$$\|x\|_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{\frac{1}{p}} \tag{2}$$

These three metrics focus on different aspects of visual significance. L_0 counts the number of pixels with different values at corresponding positions in the two images. L_2 measures the Euclidean distance between the two images x and x^*. L_∞ measures the maximum difference for all pixels at corresponding positions in the two images.

The consequences of an adversarial attack can be dramatic. For example, misclassification of a stop sign as a yield sign or a speed limit sign could lead to material and human damages. Another possible situation is when using CNNs in financial transactions and automatic bank check processing. Using handwritten character recognition algorithms to read digits from bank cheques or using neural networks for amount and signature recognition [25]. An attacker could easily fool the model to predict wrong bank account numbers or amount of money or even fake a signature. A dangerous situation, especially with such large sums of money at stake.

C. Approximate Computing

The speed of new generations of computing systems, from embedded and mobile devices to servers and computing data centers, has been drastically climbing in the past decades. This development was made possible by the advances in integrated circuits (ICs) design and driven by the increasingly high demand for performance in the majority of modern applications. However, this development is physically reaching the end of Moores law, since TSMC and Samsung are releasing 5 nm technology [26]. On the other hand, a wide range of modern applications is inherently fault-tolerant and may not require the highest accuracy. This observation has motivated the development of approximate computing (AC), which is a computing paradigm that trades power consumption with accuracy. The idea is to implement inexact/approximate computing elements that consume less energy, as far as the overall application tolerates the imprecision level in computation. This paradigm has been shown promising for inherently fault-tolerant applications such as deep/machine learning, big data analytics, and signal processing. Several AC techniques have been proposed in the literature and can be classified into three main categories based on the computing stack layer they target: software, architecture, and circuit level [27].

In this paper, we consider AC for a *totally new objective*; enhancing CNNs robustness to adversarial attacks, without losing the initial advantages of AC.

III. Threat Model

We assume an attacker attempting to conduct adversarial attacks to fool a classifier in a number of attack scenarios, which we overview in this section.

A. Transferability attacks

One of the important properties of adversarial examples is the transferability of attacks across classifiers. Specifically, transferability measures how well adversarial examples generated to target one model can also fool other related models. The property of transferability is relevant to different attack modalities. For example, in black-box scenarios [28], [29], the attacker often builds a proxy model based on observing the behavior of a target model. The attacker then creates adversarial examples using the proxy model and hopes that these attacks will transfer back to fool the original target model. This is important because the attacker may not be able to query the target model indefinitely to construct the adversarial examples. Relevant to this paper, we will also consider transferability from exact classifier to approximate classifier and vice versa.

B. Adversary knowledge (attacks scenarios)

In this work, we consider three attack scenarios:

- Transferability attack. In this attack, we assume the adversary is aware of the exact classifier internal model (its architecture and parameters). The adversary uses the exact classifier model to create adversarial examples. Therefore, we explore whether these examples transfer effectively to the approximate classifier (DA classifier).
- Black-box attack. We assume the attacker has access only to the input/output of the victim classifier (which is our approximate classifier) and has no information about its internal architecture. The adversary first uses the results of querying the victim to reverse engineer the classifier and create a substitute CNN model. With the substitute model, the attacker can attempt to generate different adversarial examples to attack the victim classifier.
- White-box attack. We assume a powerful attacker who has full knowledge of the victim classifier's architecture and parameters (including the fact that it uses approximate computing). The attacker uses this knowledge to create adversarial examples.

C. Adversarial example generation

We consider several adversarial attack generation algorithms for our attacks scenarios, including some of the most recent and potent evasion attacks. Generally, in each algorithm, the attacker tries to evade the system by adjusting malicious samples during the inference phase, assuming no influence over the training data. However, as different defenses have started to be deployed that specifically target individual adversarial attack generation strategies, new algorithms have started to be deployed that bypass these defenses. For example, methods such as defensive distillation [30] and automation detection [31] were introduced and demonstrate robustness against the Fast gradient sign attack [19]. However, the new $C\&W$ attack was able to bypass these defenses [24]. Thus, demonstrating robustness against a range of these attacks provides confidence that a defense is effective in general, against all known attack strategies, rather than against a specific strategy.

These attacks can be divided into three categories: Gradient-based attacks relying on detailed model information, including the gradient of the loss w.r.t. the input. Score-based attacks rely on the predicted scores, such as class probabilities or *logits* of the model. On a conceptual level, these attacks use the predictions to numerically estimate the gradient. Finally, decision-based attacks rely only on the final output of the model and minimizing the norm-based adversarial examples. The attacks are summarized in Table I and described in more detail in the order they appear in the table below.

TABLE I: Summary of the used attack methods. Notice that the strength estimation is based on [32].

Method	Category	Perturb. Norm	Learning	Strength
FGSM	gradient-based	L_∞	One shot	***
PGD	gradient-based	L_∞	Iterative	****
JSMA	gradient-based	L_0	Iterative	***
C&W	gradient-based	L_2	Iterative	*****
DF	gradient-based	L_2	Iterative	****
LSA	Score-based	L_2	Iterative	***
BA	Decision-based	L_2	Iterative	***
HSJ	Decision-based	L_2	Iterative	*****

Fast Gradient Sign Method (FGSM). The Fast Gradient Sign Method [19] is a single-step, gradient-based, attack. An adversarial example is generated by performing a one step gradient update along the direction of the sign of gradient at each pixel as follows:

$$x^* = x + \epsilon \cdot sign(\nabla_x \mathcal{L}_\theta(x, y)) \tag{3}$$

Where $\nabla\mathcal{L}()$ computes the gradient of the loss function $\mathcal{L}$ and θ is the set of model parameters. The $sign()$ denotes the sign function and ϵ is the perturbation magnitude.

Projected gradient descent (PGD). PGD [33] is a the strongest iterative variant of the FGSM where the adversarial example is generated as follows:

$$x^{t+1} = \mathcal{P}_{\mathcal{S}_x}(x^t + \alpha \cdot sign(\nabla_x \mathcal{L}_\theta(x^t, y))) \tag{4}$$

Where $\mathcal{P}_{\mathcal{S}_x}()$ is a projection operator projecting the input into the feasible region $\mathcal{S}_x$ and α is the added noise at each iteration. The PGD attack tries to find the perturbation that maximizes the loss of a model on a particular input while keeping the size of the perturbation smaller than a specified amount.

Jacobian based saliency map attack (JSMA). Jacobian based saliency map attack [34] builds a saliency map using the Jacobian matrix in order to define the most salient input

pixels to be modified. The Jacobian matrix of a given input x is computed as follows:

$$J_F(x) = \frac{\partial F(x)}{\partial x} = \left[\frac{\partial F_j(x)}{\partial x_i}\right]_{i \times j} \quad (5)$$

Where F denotes the second-to-last layer in [34].

Carlini & Wagner (C&W). The Carlini & Wagner attack (C&W) [35] is one of the state-of-the-art attacks. This latter has 3 forms based on different distortion measures (l_0, l_2, l_∞). In this work we only consider the l_2 form as it has the best performance. It generates adversarial examples by solving the following optimization problem:

$$\begin{aligned} \underset{\delta}{\text{minimize}} \quad & \|\delta\|_2 + c \cdot l(x+\delta) \\ \text{s.t.} \quad & x + \delta \in [0,1]^n \end{aligned} \quad (6)$$

Where $\|\delta\|_2$ is the smallest perturbation measured by the l_2 norm that makes the model missclassify into another/target class. $l(\cdot)$ is the loss function reflecting the distance between the current situation and the objective of the attack defined as:

$$l(x) = max(max_{i \neq t}\{Z(x)_i\} - Z(x)_t - \kappa) \quad (7)$$

Where $Z(x)$ is the output of the layer before the softmax called *logits*. t is the target label, and κ is called the confidence, a hyper-parameter used to enhance the transferability of the output. An adversarial example is considered as successful if $max_{i \neq t}\{Z(x)_i\} - Z(x)_t \leq 0$. In the C& W attack, the box constrained optimization problem $x+\delta \in [0,1]^n$ is turned to an unconstrained problem by replacing δ with $\frac{1}{2}(tanh(w)+1)-x$, where$w$ is a new optimizer ranging in $(-\infty, +\infty)$.

DeepFool Attack (DF). DeepFool attack [36] is an iterative attack optimized for the L_2 metric where the basic idea is to find the closest decision boundary from a clean image x in the image space, and then to go beyond that boundary to fool the classifier. It is hard to solve this problem directly in the high-dimensional and highly non-linear space in neural networks. So instead, it iteratively solves this problem with a linearized approximation. Specifically, for each iteration, it linearizes the classifier around the intermediate x^* and derives an optimal update direction on this linearized model. It then updates x^* towards this direction by a small step α. By repeating the linearize-update process until x^* crosses the decision boundary, the attack finds an adversarial example with small perturbation.

Local Search Attack (LSA). Local Search Attack [37] is an iterative attack that utilizes a local-search based technique to construct a numerical approximation to the network gradient, which is then carefully used to construct a small set of pixels in an image to perturb. This attack only uses the prediction results of the target model and does not require gradients nor probabilities.

Boundary Attack (BA). Boundary Attack [38] is an iterative algorithm based on rejective sampling algorithm in combination with a simple proposal distribution. Starting from a large perturbation sampled at each step from the proposal distribution, reducing the distance between the perturbed image and the original input. In addition to its simplicity, this attack is extremely flexible in terms of the possible adversarial criteria and has a performance similar to that of gradient-based attacks in terms of the size of minimal perturbations.

Hop Skip Jump Attack (HSJ). HSJ [39] is a powerful black-box attack that only requires final class predictions. This method builds an approximate gradient and estimates the gradient direction using binary information at the decision boundary.

IV. Defensive Approximation: Implementing Approximate CNNs

We propose to leverage approximate computing to improve the robustness of machine learning classifiers, such as CNNs, against adversarial attacks. We call this general approach Defensive Approximation (DA). In our implementation, we replace the mantissa multiplier in floating point multipliers, with a simpler approximate implementation. DA can be thought of as a perturbation-based defense [14], [13] that either add noise or otherwise filter the input data to try to interfere with any adversarial modifications to the input of a classifier. However, our approach advances the state-of-the-art by injecting perturbations throughout the classifier and directly by the hardware, thereby enhancing both robustness and performance. Moreover, the injected noise due to the approximation is input dependent, appears to sharpen the classification boundary, and increase the classification confidence. Moreover, unlike previous work, DA does not require retraining the model or pre-processing the input. In this section, we present our approximate multiplier design and analyze its properties.

A. Approximate Floating Point Multiplier

Machine learning structures such as CNNs often rely on computationally expensive operations such as convolutions that are composed of multiplications and additions. Floating-point multiplications consume most of the processing energy in both inference and training of CNNs [40], [41]. Although approximate computation can be introduced in different ways (with likely different robustness benefits), DA leverages a new approximate 32-bit floating-point multiplier, which we call *approximate floating-point multiplier* (Ax-FPM). The IEEE 754-2008 compliant floating-point format binary numbers are composed of three parts: a sign, an exponent, and a mantissa (also called fraction) [42]. The sign is the most significant bit, indicating whether the number is positive or negative. In a single-precision format, the following 8 bits represent the exponent of the binary number ranging from -126 to 127. The remaining 23 bits represent the fractional part (mantissa), normalized according to well defined rules in the standard. For most of the floating number range, the normalized format is:

$$val = (-1)^{sign} \times 2^{exp-bias} \times (1.fraction) \quad (8)$$

A floating-point multiplier (FPM) consists mainly of three units: the mantissa multiplier, the exponent adder, and the rounding unit. We choose to approximate the mantissa multiplication unit only for two main reasons: **(i)** Approximating

the mantissa prevents massive noise injection that can result from perturbing the exponent or sign, and **(ii)** The mantissa multiplication consumes 81% of the overall power of the multiplier [43].

Ax-FPM is designed based on a mantissa multiplication unit that is constructed using approximate full adders (FA). The FAs are aggressively approximated to inject computational noise within the circuit. We describe Ax-FPM by first presenting the approximate FA design, and then the approximate mantissa multiplier used to build the Ax-FPM.

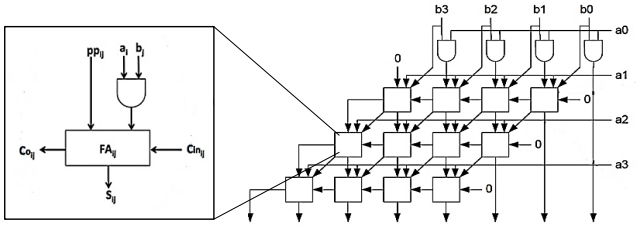

Fig. 1: An illustration of a 4×4 array multiplier architecture.

To build a power-efficient and a higher performance FPM, we propose to replace the mantissa multiplier by an approximate mantissa multiplier; an array multiplier constructed using approximate FAs. We selected an array multiplier implementation because it is considered one of the most power-efficient and high performing among conventional multiplier architectures [44]. In the array architecture, multiplication is implemented through the addition of partial products generated by multiplying the multiplicand with each bit of multiplier using AND gates, as shown in Figure 1. In the case of a 32-bit floating-point multiplier, the mantissa multiplication is a 24×24-bit multiplication.

Specifically, we build an array multiplier based on an approximate mirror adder (AMA5) [45] in place of exact FAs. The approximation of a conventional FA is performed by removing some internal circuitry, thereby resulting in power and resource reduction at the cost of introducing errors. Consider a FA with (A,B, C_{in}) as inputs and (Sum, C_{out}) as outputs (C here refers to carry). For any input combination, the logical *approximate* expressions for Sum and C_{out} are: $Sum = B$ and $C_{out} = A$ as shown in Table II. The AMA5 design is formed by only two buffers (see Figure 2), leading to the latency and energy savings relative to the exact design, but introduce errors into the computation. It is interesting to note that these errors are data dependent, appearing for specific combinations of the inputs, and ignoring the carry in value, making the injected noise difficult to predict or model.

When trying to evaluate the proposed Ax-FPM, we were interested in studying its behavior when dealing with small numbers ranging between -1 and $+1$ since most of the internal operations within CNNs are in this range. We measure the introduced error as the difference of the output of the approximate multiplier and the exact multiplier. The results are shown in Figure 3 using 100 million randomly generated multiplications across the input range from -1 to 1. Three trends

TABLE II: Truth Tables of Exact FA and AMA5.

Inputs			**Exact**		**AMA5**	
A	B	C_{in}	Sum	C_{out}	Sum	C_{out}
0	0	0	0	0	0	0
0	0	1	1	0	0	0
0	1	0	1	0	1	0
0	1	1	0	1	1	0
1	0	0	1	0	0	1
1	0	1	0	1	0	1
1	1	0	0	1	1	1
1	1	1	1	1	1	1

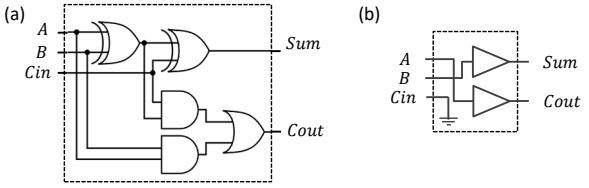

Fig. 2: Logic diagram of (a) exact Full Adder, (b) AMA5.

can be observed that will be used later to help understanding the impact of the approximation on CNN security:

(i) The first is the data-dependent discontinuity of the approximation-induced errors.

(ii) We noticed that in 96% of the cases, the approximate multiplication results in higher absolute values than the exact multiplication: For positive products, the approximate result is higher than the exact result, and for negative product results the approximate result is lower than the exact result.

(iii) In general, we notice that the larger the multiplied numbers, the larger the error magnitude added to the approximate result.

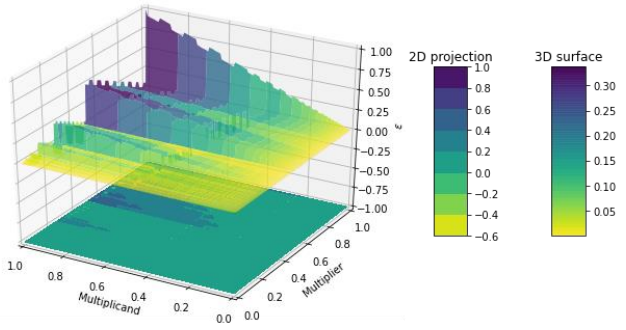

Fig. 3: Noise introduced by the approximate multiplier while the operands $\in [0, 1]$ and its projection on a 2D plan.

B. Approximate Convolution

In order to understand the impact of AC at larger scales than the individual multiplication, we track the impact on convolution operations. The approximate CNN is built using the approximate convolution operations as building blocks.

The activation functions and the pooling layers which do not use multiplication are similar to the conventional CNN. Convolution layers enable CNNs to extract data-driven features rather than relying on manual feature-extraction in classical machine learning systems. The convolution operation is performed on the layer's input data using a kernel matrix that is convoluted (piece-wise multiplied) against the input data to produce a feature map. Specifically, the output of each convolution operation is the dot product of the input matrix I and a weight kernel W computed as follows:

$$O[x][y] = B + \sum_{i=0}^{R-1} \sum_{j=0}^{R-1} I[S_x + i][S_y + j] \times W[i][j] \quad (9)$$

where O is the output feature map, B is the bias vector, R is the size of the kernel, and S is the stride.

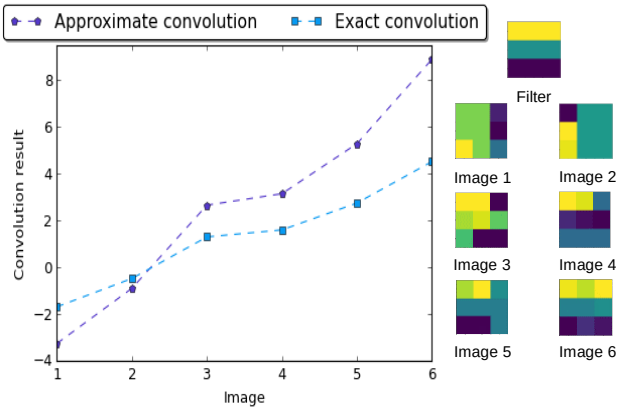

Fig. 4: Convolution result of the filter and each input image using exact and approximate convolution.

As we slide the filter over the input from left to right and top to bottom whenever the filter coincides with a **similar** portion of the input, the convolution result is high, and the neuron will fire. The weight matrix filters out portions of the input image that does not align with the filter, and the approximate multiplier helps improve this process by further increasing the output when a feature is detected. In Figure 4, we run an experiment where we choose a filter and six different images with different degrees of similarity to the chosen filter (1 to 6 from the least to the most similar), and we perform the convolution operation. We notice that the approximate convolution delivers higher results for similar inputs and lower results for dissimilar inputs. We can also notice that the higher the similarity, the higher the gap between the exact and the approximate result. Therefore, using the approximate convolution, the main features of the image that are important in the image recognition are retained and further highlighted with higher scores that will later help increase the confidence of the classification result, as explained in Section VI-B.

V. Can DA help in defending against adversarial attacks?

In this section, we empirically explore the robustness properties of DA under a number of threat models. We first explore the transferability of adversarial attacks where we evaluate whether attacks crafted for exact CNNs transfer to approximate CNNs. We then consider direct attacks against approximate CNNs in both black and white-box settings.

A. Experimental setup

The first benchmark we use is the LeNet-5 CNN architecture [46] along with the MNIST database [47], which implements a classifier for handwritten digit recognition. The MNIST consists of 60,000 training and 10,000 test images with 10 classes corresponding to digits. Each digit example is represented as a gray-scale image of 28×28 pixels, flattened as vectors of 784 features, where each feature corresponds to a pixel intensity normalized between 0 and 1. We also use the AlexNet image classification CNN [16] along with the CIFAR-10 database [48]. CIFAR-10 consists of 60,000 images, of dimension $32 \times 32 \times 3$ each. it contains ten different classes, and each class is divided into 5,000 training images and 1,000 test images. LeNet-5 consists of two convolutional layers, two max-pooling layers, and two fully connected layers. AlexNet uses five convolution layers, three max-pooling layers, and three fully connected layers. The rectified linear unit (ReLU) was used as the activation function in this evaluation, along with a dropout layer to prevent overfitting. For both models, the output layer is a special activation function called softmax that will assign probabilities to each class.

Our implementations are built using the open source machine learning framework PyTorch [49]. We use the Adam optimization algorithm to train the LeNet-5 classifier. For Alexnet, we use Stochastic Gradient Descent (SGD) with a learning rate equal to 0.01 and 0.001, respectively. Note that we do not train the approximate classifier, but rather use the same hyperparameters obtained from training the original (exact) classifier; we simply replace the multipliers with the approximate multiplier implementation.

Our reference exact CNNs are conventional CNNs based on exact convolution layers with the format Conv2d provided by PyTorch. In contrast, the approximate CNNs emulate the 32-bit Ax-FPM functionality and replace the element-wise multiplication in the convolution layers with Ax-FPM in order to assess the behavior of the approximate classifier. Except for the black-box setting where the attacker trains her own reverse-engineered proxy/substitute model, the approximate and exact classifiers share the same pre-trained parameters and the same architecture; they differ only in the hardware implementation of the multiplier.

B. Do adversarial attacks on an exact CNN transfer to an approximate CNN ?

Attack scenario. In this setting, the attacker has full knowledge of the classifier architecture and hyper-parameters, but without knowing that the model uses approximate hardware.

An example of such scenario could be in case an attacker correctly guesses the used architecture based on its widespread use for a given application (e.g., LeNet-5 in digit recognition), but is unaware of the use of DA as illustrated in Figure 5.

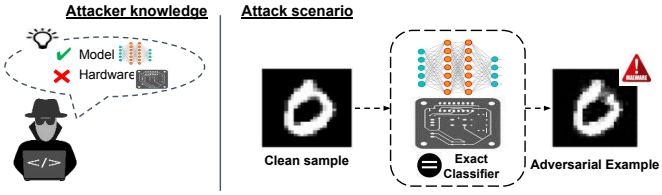

Fig. 5: Transferrability attack scenario.

Transferability Analysis. The classifier generates adversarial examples using the set of algorithms in Table I and assume that the exact classifier from Lenet-5 trained on the MNIST dataset. Notice that the hyperparameters, as well as the structure of the network, are the same between the exact and the approximate classifiers. The attacker then tests the adversarial examples against the approximate classifier. Table III presents the attacks respective success rates. We notice that the DA considerably reduces the transferability of the malicious samples and, by consequence, increases the robustness of the classifier to this attack setting. We observed that the robustness against transferability is very high, and reaches 99% for $C\&W$ attack.

TABLE III: Success rate of different attacks under exact to approximate transferability setting for MNIST.

Attack method	Exact LeNet-5	Approximate LeNet-5
FGSM	100%	12%
PGD	100%	28%
JSMA	100%	9%
C&W	100%	1%
DF	100%	17%
LSA	100%	18%
BA	100%	17%
HSJ	100%	2%

We repeat the experiment for AlexNet with CIFAR-10 dataset. For the same setting, the success of different adversarial attacks is shown in Table IV. While more examples succeed against the approximate classifier, we see that the majority of the attacks do not transfer. Thus, DA offers built-in robustness against transferability attacks.

Notice that, unlike other state-of-the-art defenses, our defense mechanism protects the network without relying on the attack details or the model specification and without any tuning or training beyond that of the original classifier. Unlike most of the perturbation-based defenses that degrade the classifiers accuracy on non-adversarial inputs, our defense strategy significantly improves the classification robustness with no baseline accuracy degradation, as we will show later in the paper.

TABLE IV: Success rate of different attacks under Exact to approximate transferability attack setting for CIFAR-10.

Attack method	Exact AlexNet	Approximate AlexNet
FGSM	100%	38%
PGD	100%	31%
JSMA	100%	32%
C&W	100%	17%
DF	100%	35%
LSA	100%	36%
BA	100%	37%
HSJ	100%	12%

C. Can we attack an approximate CNN?

In the remaining attack models, we assume that the attacker has direct access to the approximate CNN. We consider both a black-box setting where the attacker can only query the classifier and a white-box setting where a powerful attacker knows the full internal details of the approximate classifier.

Black-box Attack. In a black-box setting, the attacker has no access to the classifier architecture, parameters, and the hardware platform but can query the classifier with any input and obtain its output label. In a typical black-box attack, the adversary uses the results of many queries to the target model to reverse engineer it. Specifically, the adversary trains a substitute (or proxy) using the labeled inputs obtained from querying the original model (see Figure 6). We also conduct a black box attack on the exact classifier and evaluate how successful the black box attack is in fooling it. Essentially, we are comparing the black-box transferability of the reverse-engineered models to the original models for both the exact and the approximate CNNs.

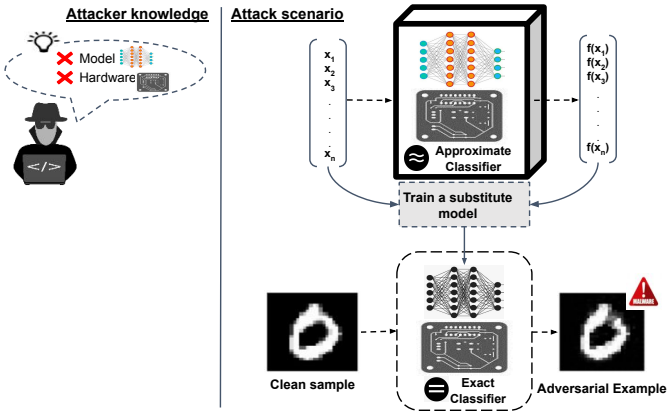

Fig. 6: Black-box attack scenario.

In Table V, we present the attack success ratios for the exact CNN and the approximate/DA CNN. DA increases resilience to adversarial attacks across various attacks and for both single-step and iterative ones: it achieves 73% classification success on adversarial examples in the worst case and the defense succeeded in up to 100% of the examples generated by C&W, PGD, and HSJ respectively.

TABLE V: Success rate of different attacks under Black-box setting for MNIST.

Attack method	Exact LeNet-5	Approximate LeNet-5
FGSM	100%	22%
PGD	100%	0%
JSMA	100%	13%
C&W	100%	0%
DF	100%	25%
LSA	100%	26%
BA	100%	27%
HSJ	100%	0%

White-box Attack. In this setting, the attacker has access to the approximate hardware along with the victim model architecture and parameters, as shown in Figure 7. In particular, the adversary has full knowledge of the defender model, its architecture and hyperparameters, the defense mechanism, along with full access to approximate gradients used to build the gradient-based attacks.

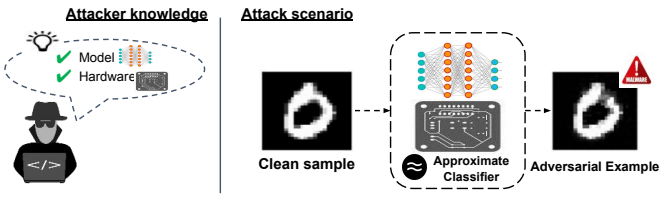

Fig. 7: White-box attack scenario.

In this scenario, we assume a powerful attacker with full access to the approximate classifier's internal model and can query it indefinitely to directly create adversarial attacks. Although DA in production would normally reduce execution time, in our experiments, we *emulate* the 32-bit Ax-FPM functionality within the approximate classifier. As a result, this makes inference extremely slow: on average, it takes 5 to 6 days to craft one adversarial example on an 8th Gen Intel core i7-8750H processor with NVIDIA GeForce GTX 1050. This led us to limit the white-box experiments; we use only two of the most efficient attacks in our benchmark: $C\&W$ and DeepFool attacks, and for a limited number of examples selected randomly from our test set.

In a white box attack, with an unconstrained noise budget, an adversary can always eventually succeed in causing an image to misclassify. Thus, robustness against this type of attack occurs in two ways: (1) The magnitude of the adversarial noise to be added: if this magnitude is high, this may exceed the ability of the attacker to interfere, or cause the attack to be easily detectable; and (2) the number of iterations, and consequently the time for producing adversarial examples: an attack that exceeds a certain limit of queries might be detected and stopped, such as the case of Cloud-based machine-learning a service platforms.

Figures 8 and 9, respectively, present different measures of L_2 for adversarial examples crafted using DF and $C\&W$ to attack both a conventional CNN and an approximate CNN.

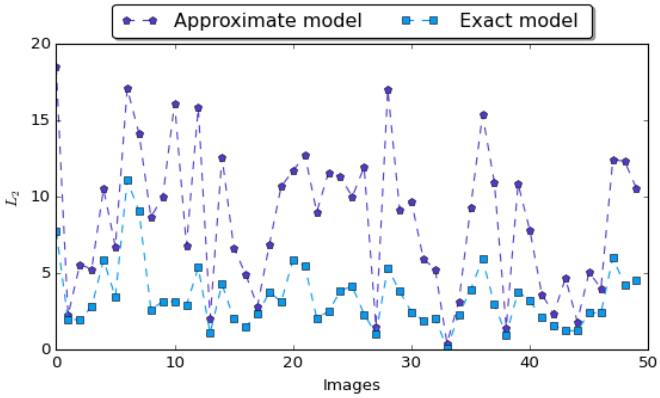

Fig. 8: L_2 values measuring distance between different clean samples from MNIST and the generated adversarial examples using DeepFool attack for approximate and exact classifiers.

We notice that the distance between a clean image and the adversarial example generated for the approximate classifier is much larger than the distance between a clean sample and the adversarial example generated for the exact classifier. On average, a difference of 5.12 for L_2-DeepFool attacks and 1.23 for L_2-C&W attack. This observation confirms that DA is more robust to adversarial perturbations since the magnitude of the adversarial noise has to be significantly higher for DF to fool DA successfully.

To understand the implication of this higher robustness in terms of observable effects on the input image, we also show the Peak Signal to Noise Ratio (PSNR) and the Mean Square Error (MSE) in Figures 10 and 11; these are two common measures of the error introduced in a reconstruction of an image. Specifically, MSE represents the average of the squares of the "errors" between the clean image and the adversarial image. The error is the amount by which the values of the original image differ from the distorted image. The PSNR is an expression for the ratio between the maximum possible value (power) of a signal and the power of distorting noise that affects the quality of its representation. It is given by the following equation: $PSNR = 20\log_{10}\left(\frac{MAX_x}{\sqrt{MSE}}\right)$. The lower the PSNR, the higher the image quality degradation is.

We notice that the adversarial examples generated on the approximate classifier have a substantially lower quality than adversarial examples generated for an exact classifier. The PSNR difference reaches $4dB$ for C&W and $7.8dB$ for DeepFool. Moreover, on average, the approximate classifier-dedicated adversarial examples have 6 times, and 3 times more MSE than the exact classifier-dedicated adversarial examples for $C\&W$ and DeepFool attacks, respectively.

We can conclude that DA provides substantial built-in robustness for all three attack models we considered. Attacks generated against an exact model do not transfer successfully to DA. Black-box attacks also achieve a low success rate against DA. Finally, even white-box attacks require substantial increases in the injected noise to fool DA. In the next section,

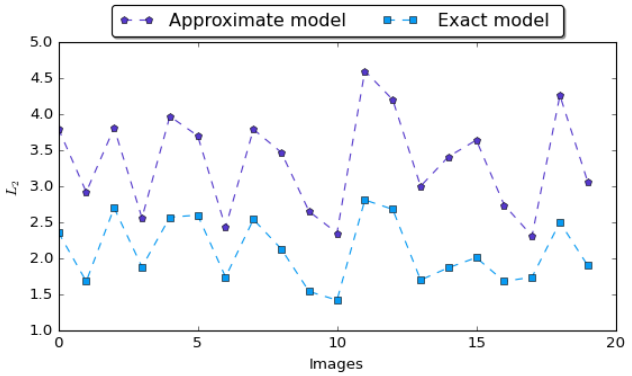

Fig. 9: L_2 values measuring distance between different clean samples from MNIST and the generated adversarial examples using C&W attack for approximate and exact classifiers.

we probe deeper into DA's internal behavior to provide some intuition and explanation for these observed robustness advantages.

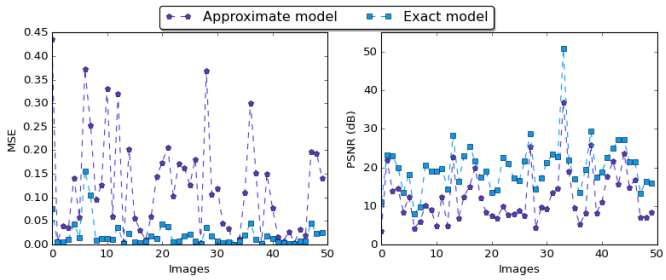

Fig. 10: MSE and PSNR values for the generated adversarial examples using DeepFool method when attacking the approximate and the exact classifiers.

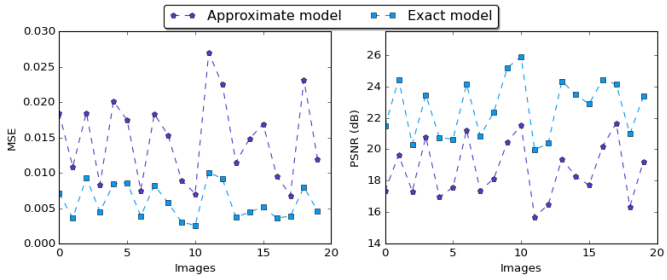

Fig. 11: MSE and PSNR values for the generated adversarial examples using L_2 C&W method when attacking the approximate and the exact classifiers.

VI. How does DA help CNN robustness?

In this section, we probe into the DA classifier's operation to attempt to explain the robustness advantages we observed empirically in the previous section. While the explainability of deep neural networks models is a known hard problem, especially under adversarial settings [50], we attempt to provide an overview of the mechanisms that we think are behind the defensive approximation impact on security. We first study the impact of the approximation on CNNs' confidence and generalization property. We follow this analysis with a mathematical argument explaining the observed robustness based on recent formulations by Lecuyer et al. [14].

A. Impact of approximation on model confidence

The output of the CNN is computed using the *softmax* function, which normalizes the outputs from the fully connected layer into a likelihood value for each output class. Specifically, this function takes an input vector and returns a non-negative probability distribution vector of the same dimension corresponding to the output classes, and whose components sum to 1. In this section, we examine the impact of approximation on the observed classifier confidence. We compare the output scores of an exact and an approximate classifier for a set of 1000 representative samples selected from the MNIST dataset: 100 randomly selected from each class. We define the classification confidence, C, as the difference between the true class l's score and the "runner-up" class score, i.e., the class with the second-highest score. C is expressed by Equation 10. The confidence ranges from 0 when the classifier gives equal likelihood to the top two or more classes, to 1 when the top class has a likelihood of 1, and all other classes 0.

$$C = output[l] - max_{j \neq l}\{output[j]\} \tag{10}$$

Figure 12 shows a scatter-plot representation of the confidence of both classifiers for the 1000 image sample set. On each point on the x-axis, we plot the confidence of that the DA classifier and the exact classifier for a particular image. It is clear the confidence of the approximate classifier is substantially higher, clustering near 1. In fact, for 97.6% of the images, DA has increased the confidence in the true class compared to other classes. We also plot the same results as a cumulative histogram in Figure 13. DA images have higher confidence; for example, in images classified by the exact classifier, less than 20% had more than 0.8 confidence. On the other hand, for the approximate classifier, 74.5% of the images reached that threshold.

Compared to the baseline feature maps generated from an exact convolution operation, for the same pre-trained weights, our approximate convolution highlights further the features. Recall that the multiplier injected noise is higher when input numbers are higher (i.e., there is a high similarity between the kernel and the input data) and lower when the inputs are lower (when the similarity is small), as shown in Figure 4. We believe that these enhanced features continue to propagate through the model resulting in a higher probability for the predicted class.

B. Theoretical Analysis

We now explore the impact of approximation on the classifier robustness leveraging some formulations from Differential Privacy (DP) [51]. Our formulation is based on recent similar

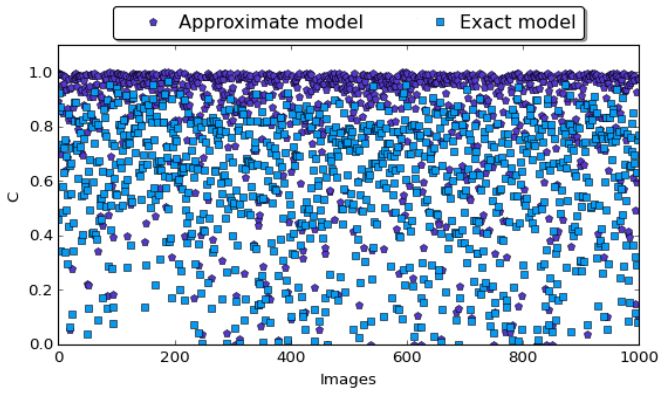

Fig. 12: Difference in confidence of exact and approximate model for clean image samples from MNIST.

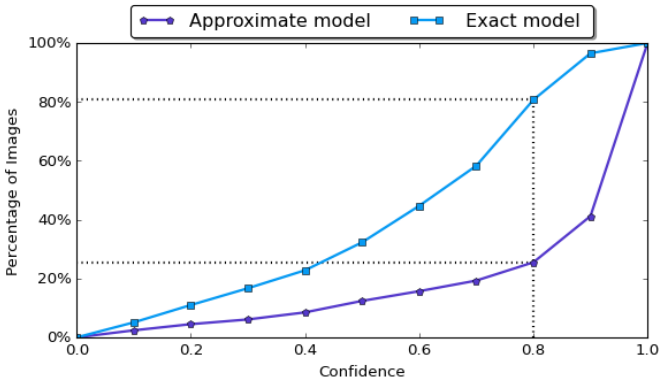

Fig. 13: Cumulative distribution of confidence.

formulations for certified robustness analysis [13], [14]. These prior works studied the impact of randomization on robustness by adding random Gaussian noise to all layers of a CNN.While the noise injected by DA follows a different distribution and is input dependent rather than stochastic, we believe that the theoretical analysis provides insights into the possible mechanisms that improve the robustness of DA.

In this analysis, we formally model the problem as follows: Let x be an input sample and h a conventional classifier. We denote $\hat{h}(x) = h(x) + \triangle(x)$ the defender implementation where $\triangle(x)$ is the data dependent randomness added by the approximate circuit. In this analysis, we use the concept of Differential Privacy (DP) [51], [52], which is concerned whether the output of a computation over a given database can reveal information about individual records in the database. To prevent such data leakage, randomness is introduced in the computation, and DP is accordingly defined as follows:

***Definition** 1:* A randomized algorithm A that takes as input a database d and outputs a value in a space O is said to satisfy (ε, δ)-DP w.r.t. a metric ρ if, for any databases d and d' with $\rho(d, d') \leqslant 1$, and for any subset of possible outputs $S \subseteq O$, we have:

$$P(A(d) \in S) \leq e^{\varepsilon} P\left(A\left(d'\right) \in S\right) + \delta \tag{11}$$

Out of this definition, Expected Output Stability Bound, a key property of DP is deduced; the expected value of an (ε, δ)-DP algorithm with bounded output is not sensitive to small changes in the input.

The intuition behind using the analogy with DP in the context of robustness to adversarial examples is to create an approximate classifier such that, given an input example, the predictions are DP with regards to the features of the input (e.g., the pixels of an image). In this setting, we can derive stability bounds for the expected output of the randomized classifier, i.e., the approximate classifier. The bounds can be interpreted as a certification for robustness to adversarial examples. In our method, the added approximate noise $\triangle(x)$ for a given input x has a similar effect to randomization.

Formally, to frame the DP into our setting, regard the feature values (e.g., pixels) of an input x as the records in a database. Let $B_p(r) := \{\alpha \in \mathbb{R}^n : \|\alpha\|_p \leq r\}$ be the p-norm ball of radius r. For a given classification model, h, and a fixed input, $x \in \mathbb{R}^n$, an attacker is able to craft a successful adversarial example of size L for a given p-norm if they find $\alpha \in B_p(L)$ such that $h(x + \alpha) \neq h(x)$. Now, consider an approximate classifier $\hat{h}$ that, on input x, outputs scores $(y_1(x), \ldots, y_K(x))$ (with $y_k(x) \in [0, 1]$ and $\sum_{k=1}^{K} y_k(x) = 1$).

Combining Lemma 1 and Corollary 1 in [14] directly implies bounds on the expected output on an approximate classifier:

***Theorem** 1:* Suppose an approximate function $\hat{h}$ satisfies (ε, δ)-DP w.r.t. a p-norm metric, and where $\hat{h}(x) = (y_1(x), \ldots, y_K(x)), y_k(x) \in [0, 1]$:

$$\forall k, \forall \alpha \in B_p(1). \hat{h}_k(x) \leq e^{\varepsilon} \hat{h}_k(x + \alpha) + \delta. \tag{12}$$

Now, consider $\alpha \in B_p(1)$, and let $x^* = x + \alpha$. From Equation 12, we have: **(i)** $\hat{h}_k(x) \leqslant e^{\varepsilon} \hat{h}_k(x^*) + \delta$, and **(ii)** $\hat{h}_{i \neq k}(x^*) \leqslant e^{\varepsilon} \hat{h}_{i \neq k}(x) + \delta$.

Practically, **(i)** gives a lower-bound on $\hat{h}_k(x^*)$ and **(ii)** gives an upper-bound on $\hat{h}_{i \neq k}(x^*)$.

These elements confirm a theoretically verifiable bounded impact of an adversarial perturbation in the input on the approximate classifier's output.

To further investigate the robustness guarantee, suppose that when the base classifier h classifies an input x the most probable class k is returned with probability p_k, and the runner-up class is returned with probability $p_{i \neq k} < p_k$. Based on [13], our approximate classifier is robust around x within the ℓ_2 radius $R = \frac{\sigma}{2}\left(\Phi^{-1}\left(p_k\right) - \Phi^{-1}\left(p_i\right)\right)$, where Φ^{-1} is the inverse of the standard Gaussian CDF. This result also holds if we replace p_k with a lower bound $\underline{p_k}$ and we replace p_i with an upper bound $\overline{p_B}$. Hence, based on Theorem 1 in [13]:

***Theorem** 2:* Let $h : \mathbb{R}^d \to \mathcal{Y}$ be any baseline classifier, and let $\varepsilon \sim \mathcal{N}\left(0, \sigma^2 I\right)$. Let $\hat{h}$ be the approximate implementation of h. Suppose $c_k \in \mathcal{Y}$ and $\underline{p_k}, \overline{p_i} \in [0, 1]$ satisfy:

$$\mathbb{P}\left(h(x+\varepsilon)=c_k\right) \geq \underline{p_k} \geq \overline{p_i} \geq \max_{c \neq c_k} \mathbb{P}(h(x+\varepsilon)=c) \quad (13)$$

Then $\hat{h}(x+\delta)=c_k$ for all $\|\delta\|_2 < R$, where

$$R = \frac{\sigma}{2}\left(\Phi^{-1}\left(\underline{p_k}\right) - \Phi^{-1}(\overline{p_i})\right) \quad (14)$$

Notice that Theorem 2 insures a certified radius R that is large when: (1) the noise level σ is high, (2) the probability of the top class c_k is high, and (3) the probability of each other class $c_{i \neq k}$ is low.

Conditions (2) and (3) of high radius basically mean a higher confidence in the sense of Equation 10. This is coherent with the results obtained in terms of robustness, given Figures 12 and 13.

The randomization technique proposed in [14] and [13] uses a totally random noise that is uncorrelated to the input. This led them to use heavy Monte Carlo simulations to estimate the randomized classifier's output, which induces a considerable time overhead limiting the applicability of these techniques, especially in time critical applications. In our approach, the noise is injected at circuit-level through the approximate design and is not artificially generated from an outside source. Moreover, the correlation to the input that highlights useful features in the convolution output gives higher confidence and thereby ensures a higher robustness radius in the sense of Theorem 2.

VII. Performance Implications

A. Impact on model Accuracy

It is important for a defense mechanism that aims to enhance robustness against adversarial attacks to keep at least an acceptable performance level for clean inputs. In fact, considerably reducing the baseline accuracy, or creating an exploding or vanishing gradient impact that makes the model sensitive to other types of noise undermines the model reliability. In our proposed approach, we maintain the same level of recognition rate even with the approximate noise in the calculations. Counter-intuitively, this data-dependent noise helps to better highlight the input's important features used in the recognition and does not affect the classification process. A drop of 0.01% in the recognition rate for the case of LeNet-5 and 1% for AlexNet is recorded as mentioned in Table VI.

TABLE VI: Accuracy results of the LeNet-5 and AlexNet CNNs.

Used Multiplier	LeNet5	AlexNet
Exact multiplier	97.78%	81%
Ax-FPM	97.77%	80%

B. Defensive approximation and sensitivity

To further investigate the stability of the approximate classifier, we explored the inference results under input random perturbation. In Table VII, random Gaussian noise ($mu = 0$, $std = 0.01, 0.05, 0.1$) was added to the input images and fed to both exact and approximate model in order to evaluate their sensitivity to random white noise (Note that noise added to a grey-scaled input image with a standard deviation equal to 0.1 is considered as aggressive noise). Measuring the accuracy of both classifiers, we notice that the approximate classifier is more robust to white Gaussian noise, and this could be explained by the fact that the introduced approximation is pushing the decision boundary further from the input image making it harder for this input to cross the boundary. Our proposed approximate classifier is not only resilient to adversarial perturbations but also to white Gaussian noise.

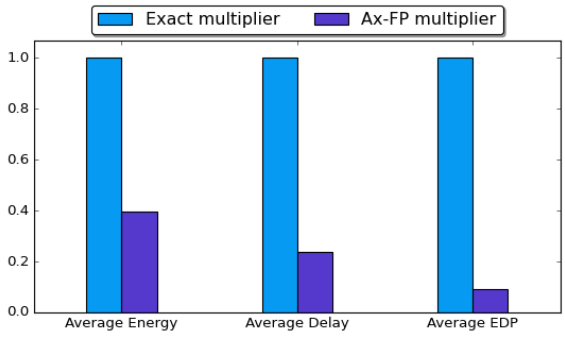

Fig. 14: Energy, delay and EDP of AMA5-based 24×24 approximate multiplier normalized to a conventional multiplier.

TABLE VII: Accuracy results when adding Gaussian noise to input images from the MNIST dataset.

Classifier	std = 0.01	std = 0.05	std = 0.1
Exact	96%	95.5%	93%
Approximate	96%	95.5%	94%

C. Impact on performance and energy consumption

This subsection shows the additional benefit of using AC, especially in the context of power-limited devices such as mobile devices, embedded devices and, Edge devices. The experiments evaluate normalized energy and delay achieved by the proposed approximate multiplier compared to a conventional baseline multiplier. Multipliers are implemented using $45\ nm$ technology via the Predictive Technology Model (PTM) using the Keysight Advanced Design System (ADS) simulation platform [53].

Figure 14 compares the energy, delay, and energy delay product (EDP) for the 24×24 approximate multiplier normalized to a conventional multiplier. The AMA5-based mantissa multiplier achieves a considerable gain in performance and energy saving, with power-saving reaching 67% of the total power consumption. The approximate circuit also achieves about 75% reduction in delay. AMA5 uses only eight transistors rather than 28, leading to a corresponding 72% savings in area.

Unlike most of the state-of-the-art defense strategies that lead to power, resource, or timing overhead, our approach results in saving energy and resources.

VIII. Discussion

This work tackles the problem of robustness to adversarial attacks from a totally new perspective: approximating the underlying hardware. DA exploits the inherent fault tolerance of deep learning systems [12], by building an approximation technique that improves robustness to adversarial noise, along with the by-product gains of AC in terms of energy and resources. Our empirical study shows promising results in terms of robustness across a wide range of attack scenarios. We notice that AC-induced noise tends to help the classifier generalize and enhances its confidence. While we do not claim a full and definitive explanation of this mechanism, we believe that the circuit-level approximate mantissa multiplier has a positive effect on the features highlighting. In fact, the AC-induced noise in the convolution layer is shown to be higher in absolute value when the inputs are similar to the convolution filter. This observation at the feature map propagates through the model and results in enhanced classification confidence, i.e., the difference between the 1^{st} class and the "runner-up". This aspect of confidence enhancement is comparable to the smoothing effect created by the randomization techniques and theoretically ensures a higher robustness guarantee.

We think that this work opens a new research direction in tackling the deep learning security problem and encourages researchers to investigate further the use of AC as a potential defense mechanism against adversarial attacks. Besides, we believe that the AC-induced noise can also be useful in a privacy preserving context.

Furthermore, we believe that the robustness enhancement introduced by hardware-supported approximation is orthogonal some other existing robustness approaches (e.g., input reprocessing and adversarial training) and hence could be used in parallel. However, further investigation needs to be done.

IX. Related Work

Several defense mechanisms were proposed to combat the effect of adversarial attacks and can be categorized as follows:

Adversarial training. Adversarial training is one of the most explored defenses against adversarial attacks. The main idea can be traced back to [19], in which models were hardened by including adversarial examples in the training data set of the model. As a result, the trained model will classify evasive samples with higher accuracy. Various attempts to combine adversarial training with other methods have resulted in better defense approaches such as cascade adversarial training [54], principled training [55]. Nonetheless, adversarial training is not effective when the attacker uses a different attack strategy than the one used to train the model [56]. Moreover, adversarial training is much more computationally intensive than training a model on the training data set only because generating evasive samples needs more computation and model fitting is more challenging (takes more epochs) [57].

Input Preprocessing: Input preprocessing depends on applying transformations to the input to remove the adversarial perturbations [58], [59]. Examples of transformation are denoising auto-encoders [60], the median, averaging, and Gaussian low-pass filters [59], and JPEG compression [58]. However, it was shown that this group of defenses is insecure under strong white-box attacks [61]; if the attacker knows the specific used transformation, the attacker can take this into account when creating the evasive sample. Furthermore, preprocessing every input requires additional computation on every input.

Gradient masking: Gradient masking relies on applying regularization to the model to make its output less sensitive to input perturbations. Papernot et al. proposed defensive distillation [30], which is based on increasing the generalization of the model by distilling knowledge out of a large model to train a compact model. Nonetheless, defensive distillation was found weak against the C&W attack [24]. Nayebi and Surya [62] purposed to use saturating networks that use a loss function that promotes the activations to be in their saturating regime. Ross and Doshi-Velez [63] proposed to regularize the gradient input by penalizing variations in the model's output with respect to changes in the input during the training of differentiable models, such as neural networks. Nonetheless, gradient masking approaches found to make white-box attacks harder and vulnerable against black-box attacks [64], [57]. Furthermore, they require re-training of pre-trained networks.

Randomization-based defenses. These techniques are the closest to our work [14], [13], [65], [66], [15]. For example, Liu et al. [66] suggest to randomize the entire DNN and predict using an ensemble of multiple copies of the DNN. Lecuyer et al. 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. Unlike our implementation, they assume introduction of random noise at different stages of the classifier. From a practical perspective, none of these works have been evaluated at scale or with realistic implementations. For example, Raghunathan et al. [15] evaluate only a tiny neural network. Other works [13], [14] consider scalability but require high overhead to implement the defense (specifically, to estimate the model output which requires running a heavy Monte Carlo simulation involving a number of different runs of the CNN). Our approach is different since not only our AC-injected noise does not require overhead but comes naturally from the simpler and faster AC implementation. Moreover, while these techniques require additional training, our implementation is a drop-in replacement of the hardware without specific training requirements, and with no changes to the architecture nor the parameters of the CNN.

X. Conclusions

To the best of our knowledge, this is the first work that proposes the use of hardware-supported approximation as a

defense strategy against adversarial attacks for CNNs. We propose a CNN implementation based on Ax-FPM, an energy-efficient approximate floating-point multiplier. While AC is used in the literature to reduce the energy and delay of CNNs, we show that AC also enhances their robustness to adversarial attacks. The proposed defense is, on average, 87% more robust against strong grey-box attacks and 87.5% against strong black-box attacks than a conventional CNN for the case of MNIST dataset, with negligible loss in accuracy. The approximate CNN achieves a significant reduction in power and EDP of 67% and 75%, respectively.

In future work, we plan to further mitigate the impact of the adversarial example; we intend to design and deploy randomly reconfigurable approximate computing elements to enhance the resiliency of the CNN. We also intend to study the generalization of our observations to more complex CNN architectures.

REFERENCES

[1] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014.

[2] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," 2016.

[3] L. Deng and Y. Liu, *Deep learning in natural language processing*. Springer, 2018.

[4] H. A. Pierson and M. S. Gashler, "Deep learning in robotics: a review of recent research," *Advanced Robotics*, vol. 31, no. 16, pp. 821–835, 2017.

[5] M. Al-Qizwini, I. Barjasteh, H. Al-Qassab, and H. Radha, "Deep learning algorithm for autonomous driving using googlenet," in *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2017, pp. 89–96.

[6] R. Miotto, F. Wang, S. Wang, X. Jiang, and J. T. Dudley, "Deep learning for healthcare: review, opportunities and challenges," *Briefings in bioinformatics*, vol. 19, no. 6, pp. 1236–1246, 2018.

[7] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," *CoRR*, vol. abs/1607.02533, 2016. [Online]. Available: http://arxiv.org/abs/1607.02533

[8] I. Evtimov, K. Eykholt, E. Fernandes, T. Kohno, B. Li, A. Prakash, A. Rahmati, and D. Song, "Robust physical-world attacks on machine learning models," *CoRR*, vol. abs/1707.08945, 2017. [Online]. Available: http://arxiv.org/abs/1707.08945

[9] V. Venceslai, A. Marchisio, I. Alouani, M. Martina, and M. Shafique, "Neuroattack: Undermining spiking neural networks security through externally triggered bit-flips," 2020.

[10] Y. Deng, "Deep learning on mobile devices - A review," *CoRR*, vol. abs/1904.09274, 2019. [Online]. Available: http://arxiv.org/abs/1904.09274

[11] M. A. Neggaz, I. Alouani, S. Niar, and F. Kurdahi, "Are cnns reliable enough for critical applications? an exploratory study," *IEEE Design Test*, pp. 1–1, 2019.

[12] M. A. Neggaz, I. Alouani, P. R. Lorenzo, and S. Niar, "A reliability study on cnns for critical embedded systems," in *2018 IEEE 36th International Conference on Computer Design (ICCD)*, 2018, pp. 476–479.

[13] J. Cohen, E. Rosenfeld, and Z. Kolter, "Certified adversarial robustness via randomized smoothing," in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. Long Beach, California, USA: PMLR, 09–15 Jun 2019, pp. 1310–1320. [Online]. Available: http://proceedings.mlr.press/v97/cohen19c.html

[14] 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.

[15] A. Raghunathan, J. Steinhardt, and P. Liang, "Certified defenses against adversarial examples," 2018.

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, ser. NIPS12. Red Hook, NY, USA: Curran Associates Inc., 2012, p. 10971105.

[17] D. C. Ciresan, U. Meier, J. Masci, and J. Schmidhuber, "Multi-column deep neural network for traffic sign classification," *Neural networks : the official journal of the International Neural Network Society*, vol. 32, pp. 333–8, 2012.

[18] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," 2013.

[19] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," 2014.

[20] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

[21] J. Lu, H. Sibai, E. Fabry, and D. Forsyth, "No need to worry about adversarial examples in object detection in autonomous vehicles," 2017.

[22] A. Boloor, X. He, C. Gill, Y. Vorobeychik, and X. Zhang, "Simple physical adversarial examples against end-to-end autonomous driving models," 2019.

[23] X. Yuan, P. He, Q. Zhu, R. R. Bhat, and X. Li, "Adversarial examples: Attacks and defenses for deep learning," *CoRR*, vol. abs/1712.07107, 2017. [Online]. Available: http://arxiv.org/abs/1712.07107

[24] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," 2016.

[25] M. B. A. Miah, M. A. Yousuf, M. S. Mia, and M. P. Miya, "Article: Handwritten courtesy amount and signature recognition on bank cheque using neural network," *International Journal of Computer Applications*, vol. 118, no. 5, pp. 21–26, May 2015, full text available.

[26] S. K. Moore, "Another step toward the end of moore's law: Samsung and tsmc move to 5-nanometer manufacturing - [news]," *IEEE Spectrum*, vol. 56, no. 6, pp. 9–10, June 2019.

[27] F. Betzel, K. Khatamifard, H. Suresh, D. J. Lilja, J. Sartori, and U. Karpuzcu, "Approximate communication: Techniques for reducing communication bottlenecks in large-scale parallel systems," *ACM Comput. Surv.*, vol. 51, no. 1, pp. 1:1–1:32, Jan. 2018. [Online]. Available: http://doi.acm.org/10.1145/3145812

[28] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," 2013.

[29] N. Papernot, P. McDaniel, and I. Goodfellow, "Transferability in machine learning: from phenomena to black-box attacks using adversarial samples," 2016.

[30] 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)*, May 2016, pp. 582–597.

[31] D. Hendrycks and K. Gimpel, "Early methods for detecting adversarial images," 2016.

[32] 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.

[33] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," 2017.

[34] N. Papernot, P. D. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," *CoRR*, vol. abs/1511.07528, 2015. [Online]. Available: http://arxiv.org/abs/1511.07528

[35] N. Carlini and D. A. Wagner, "Towards evaluating the robustness of neural networks," *CoRR*, vol. abs/1608.04644, 2016. [Online]. Available: http://arxiv.org/abs/1608.04644

[36] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," 2015.

[37] N. Narodytska and S. P. Kasiviswanathan, "Simple black-box adversarial perturbations for deep networks," *CoRR*, vol. abs/1612.06299, 2016. [Online]. Available: http://arxiv.org/abs/1612.06299

[38] W. Brendel, J. Rauber, and M. Bethge, "Decision-based adversarial attacks: Reliable attacks against black-box machine learning models," 2017.

[39] J. Chen and M. I. Jordan, "Boundary attack++: Query-efficient decision-based adversarial attack," *CoRR*, vol. abs/1904.02144, 2019. [Online]. Available: http://arxiv.org/abs/1904.02144

[40] I. Alouani, H. Ahangari, O. Ozturk, and S. Niar, "A novel heterogeneous approximate multiplier for low power and high performance," *IEEE Embedded Systems Letters*, vol. 10, no. 2, pp. 45–48, 2018.

[41] A. Guesmi, I. Alouani, M. Baklouti, T. Frikha, M. Abid, and A. Rivenq, "Heap: A heterogeneous approximate floating-point multiplier for error tolerant applications," in *Proceedings of the 30th International Workshop on Rapid System Prototyping (RSP'19)*, ser. RSP '19.

New York, NY, USA: ACM, 2019, pp. 36–42. [Online]. Available: http://doi.acm.org/10.1145/3339985.3358495

[42] "Ieee standard for floating-point arithmetic," *IEEE Std 754-2008*, pp. 1–70, 2008.

[43] J. Y. F. Tong, D. Nagle, and R. A. Rutenbar, "Reducing power by optimizing the necessary precision/range of floating-point arithmetic," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 8, no. 3, pp. 273–286, June 2000.

[44] R. Hrbacek, V. Mrazek, and Z. Vasicek, "Automatic design of approximate circuits by means of multi-objective evolutionary algorithms," in *2016 International Conference on Design and Technology of Integrated Systems in Nanoscale Era (DTIS)*, April 2016, pp. 1–6.

[45] V. Gupta, D. Mohapatra, A. Raghunathan, and K. Roy, "Low-power digital signal processing using approximate adders," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 1, pp. 124–137, Jan 2013.

[46] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov 1998.

[47] Y. LeCun and C. Cortes, "MNIST handwritten digit database," 2010. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[48] A. Krizhevsky, "Learning multiple layers of features from tiny images," Tech. Rep., 2009.

[49] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," 2019.

[50] W. Samek, *Explainable AI: interpreting, explaining and visualizing deep learning*. Springer Nature, 2019, vol. 11700.

[51] C. Dwork and A. Roth, "The algorithmic foundations of differential privacy," *Foundations and Trends in Theoretical Computer Science*, vol. 9, no. 34, pp. 211–407, 2014. [Online]. Available: http://dx.doi.org/10.1561/0400000042

[52] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor, "Our data, ourselves: Privacy via distributed noise generation," in *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006, Proceedings*, ser. Lecture Notes in Computer Science, S. Vaudenay, Ed., vol. 4004. Springer, 2006, pp. 486–503. [Online]. Available: https://doi.org/10.1007/11761679_29

[53] N. Integration and M. N. Group. (2012, Jan.) Predictive technology model (ptm) website. [Online]. Available: http://ptm.asu.edu

[54] T. Na, J. H. Ko, and S. Mukhopadhyay, "Cascade adversarial machine learning regularized with a unified embedding," 2017.

[55] A. Sinha, H. Namkoong, and J. Duchi, "Certifying some distributional robustness with principled adversarial training," 2017.

[56] P. Samangouei, M. Kabkab, and R. Chellappa, "Defense-gan: Protecting classifiers against adversarial attacks using generative models," *arXiv preprint arXiv:1805.06605*, 2018.

[57] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, "Ensemble adversarial training: Attacks and defenses," *arXiv preprint arXiv:1705.07204*, 2017.

[58] 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.

[59] 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.

[60] S. Gu and L. Rigazio, "Towards deep neural network architectures robust to adversarial examples," *arXiv preprint arXiv:1412.5068*, 2014.

[61] J. Chen, X. Wu, V. Rastogi, Y. Liang, and S. Jha, "Towards understanding limitations of pixel discretization against adversarial attacks," in *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2019, pp. 480–495.

[62] A. Nayebi and S. Ganguli, "Biologically inspired protection of deep networks from adversarial attacks," 2017.

[63] 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.

[64] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, 2017, pp. 506–519.

[65] G. S. Dhillon, K. Azizzadenesheli, Z. C. Lipton, J. Bernstein, J. Kossaifi, A. Khanna, and A. Anandkumar, "Stochastic activation pruning for robust adversarial defense," *CoRR*, vol. abs/1803.01442, 2018. [Online]. Available: http://arxiv.org/abs/1803.01442

[66] X. Liu, M. Cheng, H. Zhang, and C.-J. Hsieh, "Towards robust neural networks via random self-ensemble," 2017.