Defensive Approximation: Securing CNNs using Approximate Computing

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, we show that successful adversarial attacks against the exact classifier have poor transferability to the approximate implementation. The transferability is even poorer for the black-box attack scenarios, where adversarial attacks are generated using a proxy model. Surprisingly, the robustness advantages also apply to white-box attacks where the attacker has unrestricted access to the approximate classifier implementation: in this case, we show that substantially higher levels of adversarial noise are needed to produce adversarial examples. 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 transferability-based attacks along with up to 50% 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 4 dB degradation of the PSNR of the input image relative to the images that succeed in fooling the exact classifier.

1. Introduction

Convolutional neural networks (CNNs) and other deep learning structures provide state-of-the-art performance in many application domains, such as computer vision [62, 58], natural language processing [16], robotics [56], autonomous driving [3], and healthcare [43]. With the rapid progress in CNN's development and deployment, they are increasing concerns about their vulnerability to adversarial attacks: maliciously designed imperceptible perturbations injected within the data that cause CNNs to misclassify. Adversarial attacks have been demonstrated in real-world scenarios [31, 19, 69], making this vulnerability a serious threat to safety-critical and other applications that rely on CNNs.

Several software-based defenses have been proposed against AML attacks [52, 48, 46], but more advanced attack strategies [10, 40] also continue to evolve that demonstrate vulnerability of some of these defenses. Moreover, many of the proposed defenses introduce substantial overheads to either the training or the inference operation of CNNs [70, 46]. These overheads increase the computational requirements of these systems, which is already a significant challenge driving substantial research into algorithmic and hardware techniques to improve CNN performance and energy-efficiency [17]. Thus, finding new approaches to harden systems against AML without heavy overheads in both design-time and runtime is an area of substantial need.

In this paper, we propose a new hardware based approach to improve the robustness of machine learning (ML) classifiers. Specifically, we show that Approximate Computing (AC), designed to improve the performance and power consumption of algorithms and processing elements, can substantially improve CNN robustness to AML. Our technique, which we call *defensive approximation* (DA), substantially enhances the robustness of CNNs to adversarial attacks. 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 with full access to the internal 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 achieves similar accuracy to the exact classifier for Lenet-5 and Alexnet. DA also benefits from the conventional advantages of AC, resulting in a less complex design that is both faster and more energy efficient. Other defenses such as Defensive Quantization (DQ) [36] also provide energy efficiency benefits in addition to robustness. However, we show that because it is input-dependent DA achieves twice higher robustness to attacks than DQ.

We carry out several experiments to better understand the robustness advantages of DA. We show that the unpredictable and input-dependent variations introduced by AC improve the CNN resilience to adversarial perturbations. Experimental results show that DA has a confidence enhancement impact on non-adversarial examples; we believe that this is due to our AC multiplier which adds input dependent approximation with generally higher magnitude at large multiplication values. 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 [14], enhancing the confidence furthers the classifier's 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.

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. Subsequently, we used this approximate multiplier to implement an approximate CNN hardware accelerator (Section 4.3).
- 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 5.3.
- 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 evaluate the approximate classifiers against powerful attackers with white-box access. We observe that attackers require substantially higher adversarial perturbations to fool the approximate classifier.

We believe that DA is highly practical; it can be deployed without retraining or fine-tuning, achieving comparable classification performance to exact classifiers. In addition to security advantages, DA *improves* performance by reducing latency and energy making it an attractive choice even in Edge device settings (Appendix 8.2).

2. Background

This section first presents an overview of adversarial attacks followed by introducing AC.

2.1. 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 [13]. Despite their effectiveness and popularity, CNN-powered applications are facing a critical challenge – adversarial attacks. Many studies [65, 20, 24, 69] 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 [38] claimed that adversarial examples are not a practical threat to ML in real-life scenarios. However, physical adversarial attacks have recently been shown to be effective against CNN based applications in real-world [7].

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 $f()\ s.t.\ f(x) = l$, the problem of generating an adversarial example x^* can be formulated as a constrained optimization [71]:

$$x^* = \arg\min_{x^*} \mathscr{D}(x, x^*), s.t.\ f(x^*) = l^*,\ l \neq l^* \tag{1}$$

Where $\mathscr{D}$ 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 ($\mathscr{D}(x, x^*) < \varepsilon$ where $\varepsilon \geqslant 0$).

Distance Metrics: The adversarial examples and the added perturbations should be visually imperceptible by humans. Since it is hard to model humans' perception, researchers proposed three metrics to approximate humans' perception of visual difference, namely L_0, L_2, and L_∞ [10]. 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 [42]. 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.

2.2. 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 Moore's law, since TSMC and Samsung are releasing 5 *nm* technology [44]. 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), a computing paradigm that trades power consumption with accuracy. The idea is to implement inexact/AC 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/ML, 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 [6].

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

3. Threat Model

We assume an attacker attempting to conduct adversarial attacks to fool a classifier in a variety of attack scenarios.

3.1. Adversary knowledge (attacks scenarios)

In this work, we consider three attack scenarios:

Transferability attack. We assume the adversary is aware of the exact classifier internal model; its architecture and parameters. The adversary uses the exact classifier to create adversarial examples. Thus, 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 AC). The attacker uses this knowledge to create adversarial examples.

3.2. Adversarial example generation

We consider several adversarial attack generation algorithms for our attack 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 [52] and automation detection [25] were introduced and demonstrate robustness against the Fast gradient sign attack [20]. However, the new *C&W* attack was able to bypass these defenses [10]. 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 estimate the gradient numerically. Finally, decision-based attacks rely only on the final output of the model and minimizing the adversarial examples' norm. The attacks are summarized in Table 1.

Table 1: Summary of the used attack methods. Notice that the strength estimation is based on [2].

Method	Category	Norm	Learning	Strength
FGSM [20]	gradient-based	L_∞	One shot	***
PGD [41]	gradient-based	L_∞	Iterative	****
JSMA [54]	gradient-based	L_0	Iterative	***
C&W [10]	gradient-based	L_2	Iterative	*****
DF [45]	gradient-based	L_2	Iterative	****
LSA [47]	Score-based	L_2	Iterative	***
BA [8]	Decision-based	L_2	Iterative	***
HSJ [11]	Decision-based	L_2	Iterative	*****

4. Defensive Approximation: Implementing Approximate CNNs

We propose to leverage approximate computing to improve the robustness of ML classifiers, such as CNNs, against adversarial attacks. We call this general approach **Defensive Approximation** (DA). The closest approach to DA is the perturbation-based defense [34, 14] that either adds 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 approximate hardware, thereby enhancing both robustness and power efficiency.

Technically, the approximate design process is driven by two main considerations:

(a) Injecting significant noise that can influence the CNN behavior, and allows by-product power gains.

(b) Keeping the cumulative noise magnitude under control to avoid impacting the baseline accuracy of the CNN.

For consideration (b), we purpose to an implementation that replaces the conventional mantissa multiplier in floating point multipliers, with a simpler approximate design. This choice is backed by the study in [49, 50], which show that errors in the exponent part might have a drastic impact on CNNs accuracy. Consideration (a) means that the approximate design needs to be aggressive to induce significant noise. Therefore, as detailed in the next subsection, we chose the corner case, i.e., the most aggressive approximate design [23, 22], and used it to replace the mantissa computation logic in a basic floating point multiplier.

In this section, we present our approximate multiplier design and analyze its properties.

4.1. Approximate Floating Point Multiplier

ML structures, such as CNNs, often rely on computationally expensive operations, e.g., convolutions that are composed of multiplications and additions. Floating-point multiplications consume most of the processing energy in both inference and training of CNNs [5, 22]. 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) [1]. 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). For most of the floating number range, the normalized format is:

$$val = (-1)^{sign} \times 2^{exp-bias} \times (1.fraction) \tag{3}$$

A floating-point multiplier (FPM) consists mainly of three units: mantissa multiplier, exponent adder, and a rounding unit. The mantissa multiplication consumes 81% of the overall power of the multiplier [67].

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.

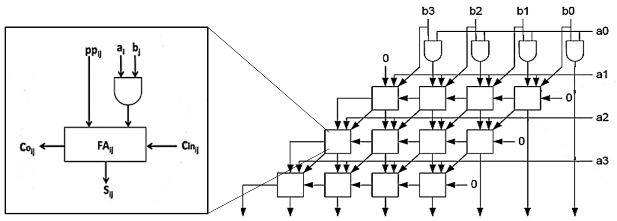

Figure 1: An illustration of a 4×4 array multiplier architecture.

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 among conventional multiplier architectures [26]. 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.

Specifically, we build an array multiplier based on an approximate mirror adder (AMA5) [23] 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 $C_{out} = A$. The AMA5 design is constituted by only two buffers (see Figure 2), leading to the latency and energy savings relative to the exact design, but more importantly, introduce errors into the computation. It is worth noting 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.

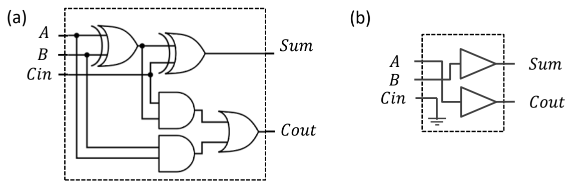

Figure 2: Logic diagram of (a) exact Full Adder, (b) AMA5 (used in this work).

When trying to evaluate the proposed Ax-FPM, we were interested in studying its behavior under small input 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 can be observed that will be used later to help understanding the impact of the approximation using Ax-FPM 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, and **(iii)** In general, we notice that the larger the multiplied numbers, the larger the error magnitude added to the approximate result. As shown later, these observations will help understand the mechanism that we think is behind robustness enhancement.

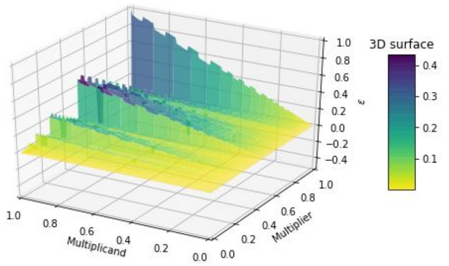

Figure 3: Noise introduced by the approximate multiplier while the operands $\in [0,1]$.

4.2. 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 ML 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.

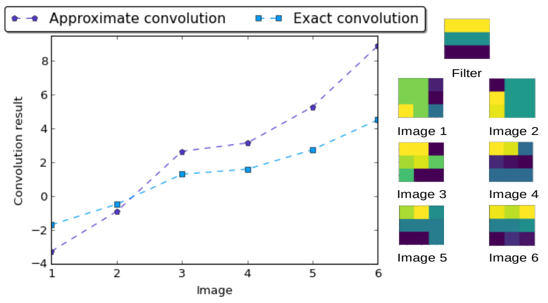

Figure 4: Convolution result of the filter and each input image using exact and approximate convolution.

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 using Ax-FPM 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 Ax-FPM approximation results. Therefore, by 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 6.

4.3. AC Design Space Exploration

For the sake of comparison, we proceed to a design space exploration considering multipliers' accuracy and resource utilization as optimization objectives. The comparison of our design with the optimal approximate design given by the exploration and referred to as HEAP led to a clearly dominant choice in favor of our design given the following perspectives:

(i) Resource and power efficiency: Given its aggressive design, AxFPM achieves the lowest power consumption and resource utilization.

(ii) Robustness: While other approximate designs had a positive impact on CNNs robustness, we noticed that AxFPM achives the highest enhancement.

(iii) Accuracy: CNNs are highly approximation-tolerant, and even aggressive multiplier design didn't impact overall accuracy. AxFPM, as well as HEAP have insignificant impact on accuracy.

While these observations give an insight on the potential generalizability of AC positive impact, AxFPM is a non dominated design outperforming other design in both robustness, power consumption and resource utilization, while having the same overall accuracy level. Hence, in the remainder of the paper, we will focus only on Ax-FPM based DA. Further details and results can be found in the Appendix.

5. 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.

5.1. Experimental setup

The first benchmark we use is the LeNet-5 CNN architecture [32] along with the MNIST database [33], 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, where each feature corresponds to a pixel intensity normalized between 0 and 1. We also use the AlexNet image classification CNN [30] along with the CIFAR-10 database [29]. CIFAR-10 consists of 60,000 images, of dimension $32 \times 32 \times 3$ each and contains ten different classes. 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 ML framework PyTorch [55]. 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 *no retraining* is applied in DA, we rather use the same hyper-parameters obtained from the original (exact) classifier; we simply replace the multipliers with the approximate multiplier.

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 multiplication in the convolution layers with Ax-FPM in order to assess the behavior of the approximate classifier. Since we are simulating a cross-layer (approximate) implementation from gate-level up to system-level, the experiments (forward and backward gradient) are highly time consuming, which limited our experiments to the two datasets MNIST and CIFAR-10. Except for the black-box setting where the attacker trains his 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.

5.2. 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.

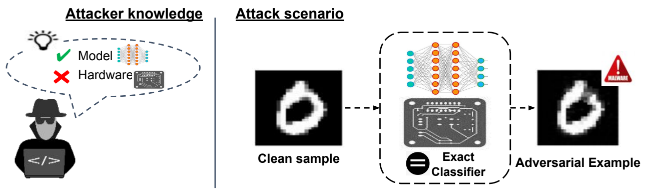

Figure 5: Transferrability attack scenario.

Transferability Analysis. The classifier generates adversarial examples using the set of algorithms in Table 1 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 2 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 2: Attacks transferability success rates 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 3. 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.

Table 3: Attacks transferability success rates 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%

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 training beyond that of the original classifier. Unlike most of the perturbation-based defenses that degrade the classifier's accuracy on non-adversarial inputs, our defense strategy significantly improves the classification robustness with no baseline accuracy degradation, as we will show in Section 8.1.

5.3. 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 black-box and white-box settings.

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.

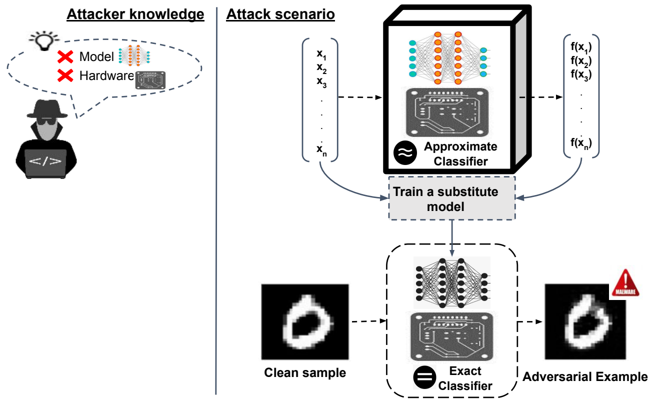

Figure 6: Black-box attack scenario.

In Table 4, 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 4: Black-box attacks success rates 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, the defense mechanism, along with full access to approximate gradients used to build the gradient-based attacks. In essence, the attacker is aware of our defense, and can adapt around it with full knowledge of the model and the hardware, which is a recommended methodology for evaluating new defenses [9].

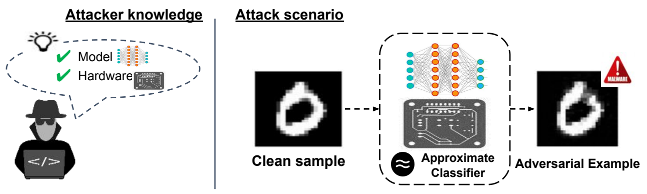

Figure 7: White-box attack scenario.

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 for different classes.

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 through 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.

Figures 8 and 9, respectively, present different measures of L_2 for adversarial examples crafted using DF and *C&W* attacking both a conventional CNN and an approximate CNN. We notice that the distance between a clean image and the adversarial example generated under DA 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

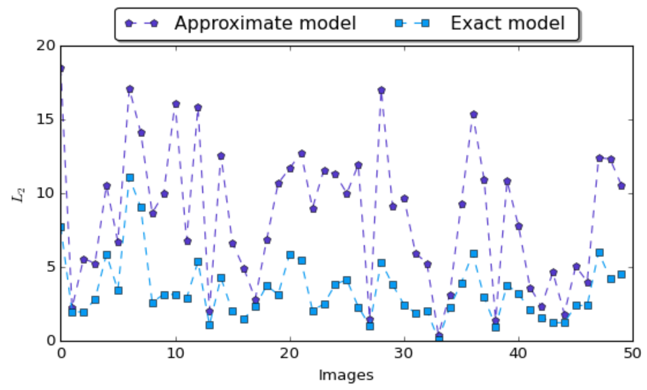

Figure 8: Required perturbations by DeepFool attack, measured using L_2 distance, to generate MNIST adversarial examples for approximate and exact classifiers.

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. 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 for DA is more noisy 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 DA-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.

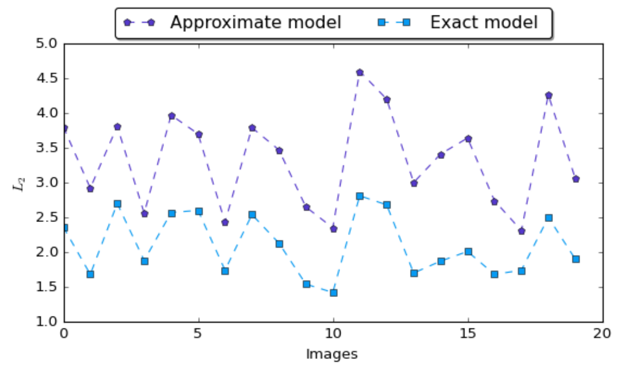

Figure 9: Required perturbations by C&W attack, measured using L_2 distance, to generate MNIST adversarial examples for approximate and exact classifiers.

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

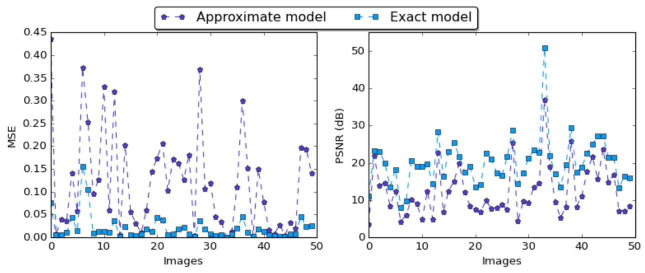

Figure 10: MSE and PSNR values for the generated adversarial examples using DeepFool method when attacking the approximate and the exact classifiers.

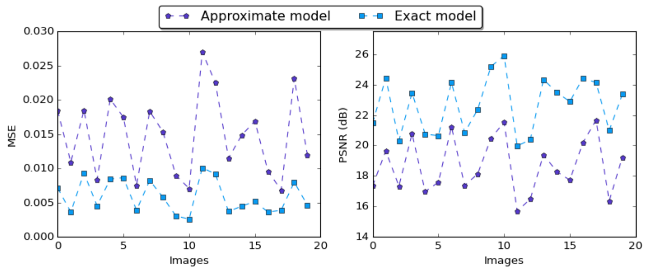

Figure 11: MSE and PSNR values for the generated adversarial examples using L_2 C&W method when attacking the approximate and the exact classifiers.

6. 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 [61], we attempt to provide an overview of the mechanisms that we think are behind the DA impact on security. We study the impact of the approximation on CNNs' confidence and generalization property. We follow this analysis in the Appendix with a mathematical argument explaining the observed robustness based on recent formulations by Lecuyer et al. [34].

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

We plot the cumulative distribution of confidence for both classifiers in Figure 12. DA images have higher confidence; for example, in images classified by the exact classifier, less than 20% had higher than 0.8 confidence. On the other hand, for the approximate classifier, 74.5% of the images reached that threshold.

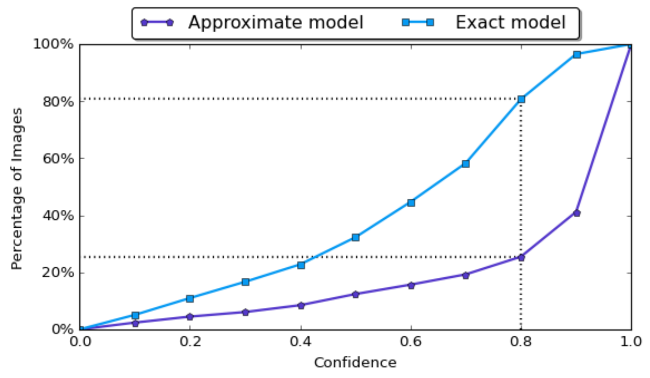

Figure 12: Cumulative distribution of confidence.

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. This higher confidence requires thereby higher noise to decrease the true label's likelihood, and increase another label's. This aspect can be theoretically modeled as detailed in the Appendix.

7. How does DA compare to other reduced precision techniques?

In this section, we investigate the impact of other reduced precision techniques on robustness. We first compare DA to DQ, and then study the impact of using Bfloat16 data representation on the system performance and robustness.

7.1. Defensive Quantization

Defensive quantization [36] was proposed to jointly optimize the efficiency and robustness of deep learning models. A 4-bit quantized model was trained on CIFAR-10 using the Dorefa-Net method [72]. The model architecture is detailed in Appendix B. We consider two ways of quantization: (i) Weight quantization, where only the weights are quantized, and (ii) Full quantization where the weights of each convolutional and dense block and the output of each activation function are quantized. In Table 5, we report transferability between exact (32-bit floating point), approximate model (using DA), fully quantized and weight-only quantized model. We notice that DA is almost two times more robust against transferability attacks than DQ under FGSM, PGD and C&W attacks. We believe that this is due to the difference in terms of noise distribution between DA and DQ. In fact, while DQ-induced noise tends to make the initial decision boundary smoother, the input-dependent noise in DA makes its decision boundary randomly different from the initial model.

Table 5: Comparing attacks transferability success rates for CIFAR-10 when using DA and DQ.

Attack method	Exact	DA:	DQ: Full	DQ: Weight-only
FGSM	100%	38%	60%	61%
PGD	100%	31%	74%	73%
C&W	100%	17%	68%	68%

7.2. BFloat16

The Bfloat16 (Brain Floating Point) [28] is a truncated version of the 32-bit IEEE 754 single-precision floating-point format (float32). It is composed of one sign bit, eight exponent bits, and seven mantissa bits giving the range of a full 32-bit number but in the data size of a 16-bit number. Bfloat16 is used in machine learning applications and intended to reduce the storage requirements and increase computing speed without losing precision. To evaluate the impact of Bfloat16 on deep neural networks robustness, we use Pytorch framework [55] to implement Bfloat16-based CNN architectures and test them for MNIST and CIFAR-10 benchmarks. We notice that using Bfloat16 achieves the same prediction accuracy as the full precision 32-bit floating point. No remarkable change was noticed at the output of the convolutional layers nor in the confidence of the models. We believe this is due to the nature of the noise introduced by reducing data representation accuracy. A more detailed discussion can be found in Section 9. Figure 13 shows the noise resulting from multiplying 100 million randomly generated Bfloat16 numbers compared to their corresponding 32-bit floating point. This figure shows that, in contrast with our approximate multiplier (Figure 3), Bfloat16 multiplication results in mostly negative noise with orders of magnitude lower than DA-induced noise. Moreover, the Bfloat16 noise is not input-dependent, and has no specific impact on the model confidence.

Accordingly, no improvement in the robustness was noticed when attacking the Bfloat16 model using FGSM, PGD and *C&W*.

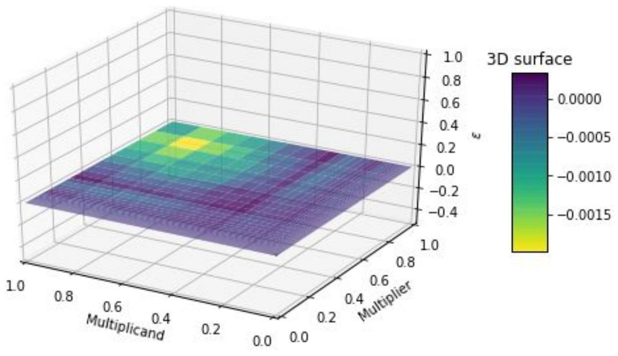

Figure 13: Noise introduced by multiplying bfloat16 while the operands $\in [0, 1]$.

8. Baseline Performance Implications

8.1. 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 8.

Table 6: Accuracy results of Float32, approximate model, fully quantized, weight-only quantized and Bfloat16 models.

Used Multiplier	MNIST	CIFAR-10
Float32	97.93%	81%
Approximate (DA)	97.67%	80%
Fully quantized	-	80%
Weight-only quantized	-	80%
Bfloat16	97.93	81%

8.2. Impact on performance and energy consumption

Here we show the additional benefit of using AC, especially in the context of power-limited devices, such as mobile, embedded, 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 [27].

Table 7: Energy and delay Comparison.

Multiplier	Average energy	Average delay
Exact multiplier	1	1
Ax-FPM	0.487	0.29
Bfloat16	0.4	0.4

Table 7 compares the energy and delay for the approximate multiplier and the Bfloat16 multiplier normalized to a conventional multiplier.

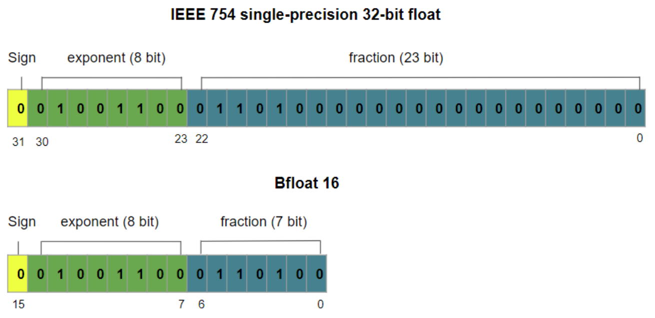

Figure 14: Contrast between IEEE 754 Single-precision 32-bit floating-point format and Bfloat16 format.

Bfloat16 data representation is shown in Figure 14, with 1 sign bit, a 8-bit exponent and a 7-bit fraction. The used Bfloat16 multiplier has similar architecture to that of Ax-FPM. However for the mantissa multiplier we use the conventional Booth multiplier instead of the array multiplier. Ax-FPM achieves up to 51% and 71% savings in energy and delay, respectively compared to the baseline multiplier. Bfloat16 implementation results in comparable savings with around 60% lower power and delay. However, this comes without robustness advantages, as mentioned earlier. Unlike most of the state-of-the-art defense strategies that lead to power, resource, or timing overhead, DA results in saving energy and area.

9. Discussion

This work tackles the problem of robustness to adversarial attacks from a new perspective: approximation in the underlying hardware. DA exploits the inherent fault tolerance of deep learning systems [49] to provide resilience while also obtaining 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 perturbations tend to help the classifier generalize and enhances its confidence. We believe that this observation is possibly due to the specific AC multiplier we used where the introduced noise is input-dependent and non-uniform. When we observe the effect on the convolution layer, we see higher absolute values 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} and runner-up classes. This aspect of confidence enhancement resembles the smoothing effect observed by some recently proposed randomization techniques [34].

We believe that our study makes important contributions in demonstrating the general potential of approximate computing in this new dimension. However, we believe that substantial further research remains which we hope to tackle in our future work: (1) More work is needed to carefully understand the relationship between the patterns of induced noise and the observed robustness to adversarial attacks to guide the selection of approximation approaches; (2) We would also like to explore whether there is additional protection that results from adapting the approximation function over time; (3) We believe that DA is orthogonal to some of the other AML defenses and, deployed together, they may result in even higher protection against AML; (4) Some AML defenses unintentionally make the model more susceptible to privacy related attacks [64]; we believe that DA does not have a similar effect and would like to study its implication on privacy-preserving in the future; and (5) We would like to explore DA in the context of other learning structures beyond CNNs.

DA has two important advantages compared to DQ: (1) DA results in input-dependent noise, while DQ results in a deterministic network that can be efficiently reverse engineered and undermined by adaptive white-box attacks; (2) DQ requires retraining/fine-tuning the model to avoid drastic accuracy drop, while DA does not require retraining.

Compared to DA, when proceeding to Bfloat16 quantization, we did not notice any improvement in the model robustness, which could be explained by the fact that Bfloat16 results in uniform low-amplitude noise distribution that is not sufficient to impact CNNs behavior.

While we considered full precision floating point CNNs, we believe that DA can also apply to quantized and sparse networks [4, 39, 66] with similar impact on security. Prior work [50] shows that quantized networks tolerate errors, implying that DA can potentially be deployed without degrading accuracy.

Our experimental setup is based on simulating a cross-layer implementation from gate-level up to system-level. Therefore, the experiments are computationally and time demanding, which limited our experiments to the two datasets MNIST and CIFAR-10. This limitation was the same for techniques that require Monte Carlo Simulations such as [34]. Our observations held for 5-layer-CNN (LeNet) and 8-layer-CNN (AlexNet). In future work, we are planning to solve the simulation time limitation by a real hardware implementation, which facilitate evaluating DA for larger networks.

10. Related Work

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

Adversarial training (AT). AT is one of the most explored defenses against adversarial attacks. The main idea can be traced back to [20], in which models were hardened by including adversarial examples in the training data set of the model. As a result, the trained model classifies evasive samples with higher accuracy. Various attempts to combine AT with other methods have resulted in better defense approaches such as cascade adversarial training [46], principled training [63]. Nonetheless, AT is not effective when the attacker uses a different attack strategy than the one used to train the model [60]. 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) [68].

Input Preprocessing: Input preprocessing depends on applying transformations to the input to remove the adversarial perturbations [15, 51]. Examples of transformation are denoising auto-encoders [21], the median, averaging, and Gaussian low-pass filters [51], and JPEG compression [15]. However, it was shown that these defenses are insecure under strong white-box attacks [12]; if the attacker knows the specific used transformation, it can be taken into account when creating the attack. Furthermore, preprocessing requires additional computation on every input.

Gradient Masking (GM): GM relies on applying regularization to the model to make its output less sensitive to input perturbations. Papernot et al. proposed defensive distillation [52], which is based on increasing the generalization of the model by distilling knowledge out of a large model to train a compact model. Nonetheless, distillation was found weak against *C&W* attack [10]. Nayebi and Surya [48] purposed to use saturating networks that use a loss function that promotes the activations to be in their saturating regime. [59] 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. Nonetheless, GM approaches found to make white-box attacks harder and vulnerable against black-box attacks [53, 68]. Furthermore, they require re-training of pre-trained networks.

Randomization-based defenses. These techniques are the closest to our work [34, 14, 18, 37, 57]. Liu et al. [37] suggest to randomize the entire DNN and predict using an ensemble of multiple copies of the DNN. Lecuyer et al. [34] 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. From a practical perspective, none of these works has been evaluated at scale or with realistic implementations. For example, Raghunathan et al. [57] evaluate only a tiny neural network. Other works [14, 34] 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 noise does not require overhead but comes naturally from the simpler and faster AC implementation. Moreover, while these techniques require additional training, DA 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.

11. 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 delay of 50% and 67%, respectively.

Acknowledgements

We would like to thank the reviewers and shepherd for the feedback and suggestions that have substantially improved the paper. This work was partially supported by NSF grants CNS-1646641, CNS-1619322 and CNS-1955650.

References

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

[2] N. Akhtar and A. Mian. Threat of adversarial attacks on deep learning in computer vision: A survey. *IEEE Access*, 6:14410–14430, 2018.

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

[4] Karim M. A. Ali, Ihsen Alouani, Abdessamad Ait El Cadi, Hamza Ouarnoughi, and Smail Niar. Cross-layer cnn approximations for hardware implementation. In Fernando Rincón, Jesús Barba, Hayden K. H. So, Pedro Diniz, and Julián Caba, editors, *Applied Reconfigurable Computing. Architectures, Tools, and Applications*, pages 151–165, Cham, 2020. Springer International Publishing.

[5] I. Alouani, H. Ahangari, O. Ozturk, and S. Niar. A novel heterogeneous approximate multiplier for low power and high performance. *IEEE Embedded Systems Letters*, 10(2):45–48, 2018.

[6] Filipe Betzel, Karen Khatamifard, Harini Suresh, David J. Lilja, John Sartori, and Ulya Karpuzcu. Approximate communication: Techniques for reducing communication bottlenecks in large-scale parallel systems. *ACM Comput. Surv.*, 51(1):1:1–1:32, January 2018.

[7] Adith Boloor, Xin He, Christopher Gill, Yevgeniy Vorobeychik, and Xuan Zhang. Simple physical adversarial examples against end-to-end autonomous driving models, 2019.

[8] Wieland Brendel, Jonas Rauber, and Matthias Bethge. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models, 2017.

[9] Nicholas Carlini, Anish Athalye, Nicolas Papernot, Wieland Brendel, Jonas Rauber, Dimitris Tsipras, Ian Goodfellow, Aleksander Madry, and Alexey Kurakin. On evaluating adversarial robustness, 2019.

[10] Nicholas Carlini and David A. Wagner. Towards evaluating the robustness of neural networks. *CoRR*, abs/1608.04644, 2016.

[11] Jianbo Chen and Michael I. Jordan. Boundary attack++: Query-efficient decision-based adversarial attack. *CoRR*, abs/1904.02144, 2019.

[12] Jiefeng Chen, Xi Wu, Vaibhav Rastogi, Yingyu Liang, and Somesh Jha. Towards understanding limitations of pixel discretization against adversarial attacks. In *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 480–495. IEEE, 2019.

[13] Dan C. Ciresan, Ueli Meier, Jonathan Masci, and Jürgen Schmidhuber. Multi-column deep neural network for traffic sign classification. *Neural networks : the official journal of the International Neural Network Society*, 32:333–8, 2012.

[14] Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. Certified adversarial robustness via randomized smoothing. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 1310–1320, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

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

[16] Li Deng and Yang Liu. *Deep learning in natural language processing*. Springer, 2018.

[17] Yunbin Deng. Deep learning on mobile devices - A review. *CoRR*, abs/1904.09274, 2019.

[18] Guneet S. Dhillon, Kamyar Azizzadenesheli, Zachary C. Lipton, Jeremy Bernstein, Jean Kossaifi, Aran Khanna, and Anima Anandkumar. Stochastic activation pruning for robust adversarial defense. *CoRR*, abs/1803.01442, 2018.

[19] Ivan Evtimov, Kevin Eykholt, Earlence Fernandes, Tadayoshi Kohno, Bo Li, Atul Prakash, Amir Rahmati, and Dawn Song. Robust physical-world attacks on machine learning models. *CoRR*, abs/1707.08945, 2017.

[20] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples, 2014.

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

[22] Amira Guesmi, Ihsen Alouani, Mouna Baklouti, Tarek Frikha, Mohamed Abid, and Atika 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)*, RSP '19, pages 36–42, New York, NY, USA, 2019. ACM.

[23] 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*, 32(1):124–137, Jan 2013.

[24] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[25] Dan Hendrycks and Kevin Gimpel. Early methods for detecting adversarial images, 2016.

[26] 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)*, pages 1–6, April 2016.

[27] Nanoscale Integration and Modeling (NIMO) Group. Predictive technology model (ptm) website, January 2012.

[28] Dhiraj Kalamkar, Dheevatsa Mudigere, Naveen Mellempudi, Dipankar Das, Kunal Banerjee, Sasikanth Avancha, Dharma Teja Vooturi, Nataraj Jammalamadaka, Jianyu Huang, Hector Yuen, Jiyan Yang, Jongsoo Park, Alexander Heinecke, Evangelos Georganas, Sudarshan Srinivasan, Abhisek Kundu, Misha Smelyanskiy, Bharat Kaul, and Pradeep Dubey. A study of bfloat16 for deep learning training, 2019.

[29] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[30] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, page 1097–1105, Red Hook, NY, USA, 2012. Curran Associates Inc.

[31] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *CoRR*, abs/1607.02533, 2016.

[32] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov 1998.

[33] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.

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

[35] J. Liang, J. Han, and F. Lombardi. New metrics for the reliability of approximate and probabilistic adders. *IEEE Transactions on Computers*, 62(9):1760–1771, Sep. 2013.

[36] Ji Lin, Chuang Gan, and Song Han. Defensive quantization: When efficiency meets robustness, 2019.

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

[38] Jiajun Lu, Hussein Sibai, Evan Fabry, and David Forsyth. No need to worry about adversarial examples in object detection in autonomous vehicles, 2017.

[39] Yufei Ma, N. Suda, Yu Cao, J. Seo, and S. Vrudhula. Scalable and modularized rtl compilation of convolutional neural networks onto fpga. In *2016 26th International Conference on Field Programmable Logic and Applications (FPL)*, pages 1–8, Aug 2016.

[40] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks, 2017.

[41] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks, 2017.

[42] Mohammad Badrul Alam Miah, Mohammad Abu Yousuf, Md. Sohag Mia, and Md. Parag Miya. Article: Handwritten courtesy amount and signature recognition on bank cheque using neural network. *International Journal of Computer Applications*, 118(5):21–26, May 2015. Full text available.

[43] Riccardo Miotto, Fei Wang, Shuang Wang, Xiaoqian Jiang, and Joel T Dudley. Deep learning for healthcare: review, opportunities and challenges. *Briefings in bioinformatics*, 19(6):1236–1246, 2018.

[44] S. K. Moore. Another step toward the end of moore's law: Samsung and tsmc move to 5-nanometer manufacturing - [news]. *IEEE Spectrum*, 56(6):9–10, June 2019.

[45] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks, 2015.

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

[47] Nina Narodytska and Shiva Prasad Kasiviswanathan. Simple black-box adversarial perturbations for deep networks. *CoRR*, abs/1612.06299, 2016.

[48] Aran Nayebi and Surya Ganguli. Biologically inspired protection of deep networks from adversarial attacks, 2017.

[49] 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)*, pages 476–479, 2018.

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

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

[52] 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)*, pages 582–597, May 2016.

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

[54] Nicolas Papernot, Patrick D. McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. *CoRR*, abs/1511.07528, 2015.

[55] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019.

[56] Harry A Pierson and Michael S Gashler. Deep learning in robotics: a review of recent research. *Advanced Robotics*, 31(16):821–835, 2017.

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

[58] Joseph Redmon and Ali Farhadi. Yolo9000: Better, faster, stronger, 2016.

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

[60] Pouya Samangouei, Maya Kabkab, and Rama Chellappa. Defense-gan: Protecting classifiers against adversarial attacks using generative models. *arXiv preprint arXiv:1805.06605*, 2018.

[61] Wojciech Samek. *Explainable AI: interpreting, explaining and visualizing deep learning*, volume 11700. Springer Nature, 2019.

[62] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2014.

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

[64] Liwei Song, Reza Shokri, and Prateek Mittal. Privacy risks of securing machine learning models against adversarial examples. *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, Nov 2019.

[65] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks, 2013.

[66] Enzo Tartaglione, Skjalg Lepsø y, Attilio Fiandrotti, and Gianluca Francini. Learning sparse neural networks via sensitivity-driven regularization. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 3878–3888. Curran Associates, Inc., 2018.

[67] 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*, 8(3):273–286, June 2000.

[68] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. *arXiv preprint arXiv:1705.07204*, 2017.

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

[70] Cihang Xie, Yuxin Wu, Laurens van der Maaten, Alan L Yuille, and Kaiming He. Feature denoising for improving adversarial robustness. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 501–509, 2019.

[71] Xiaoyong Yuan, Pan He, Qile Zhu, Rajendra Rana Bhat, and Xiaolin Li. Adversarial examples: Attacks and defenses for deep learning. *CoRR*, abs/1712.07107, 2017.

[72] Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients, 2018.

Appendices

A. Appendix: Design Space Exploration

Accuracy HEAP [22] is the result of a design space exploration among combinations of different approximate full adders leading to minimal accuracy loss. The selection process of optimal circuit design was performed after an exhaustive accuracy evaluation of all possible configurations.The metrics used to evaluate the accuracy of different combinations are the mean relative error distance (MRED) [35] $MRED = \frac{1}{n}\sum_{i=1}^{n}\frac{|\hat{Y}-Y|}{Y}$, and the normalized mean error distance (NMED): $NMED = \frac{1}{n}\sum_{i=1}^{n}\frac{|\hat{Y}-Y|}{P_{max}}$. Where Y is the exact result, $\hat{Y}$ is the approximate result, and P_{max} is the maximum product.

Table 8: Accuracy results of the LeNet-5 CNN and different multipliers.

Multiplier	CNN Accuracy	MRED	NMED
Exact multiplier	97.93%	0	0
HEAP [22]	97.86%	0.12	0.03
Ax-FPM	97.67%	0.33	0.08

Table 8 includes the accuracy results of the exact multiplier and the two approximate multipliers. Ax-FPM is substantially less accurate than HEAP. However, when we evaluate CNN accuracy (using the recognition rate), we notice a negligible accuracy loss for both approximate multipliers, confirming the inherent error-resiliency of CNNs.

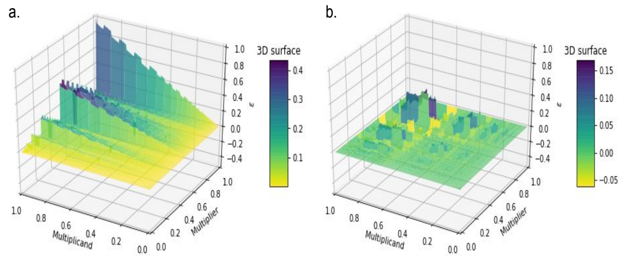

Figure 15: Noise introduced by the approximate multiplier while the operands $\in [0,1]$ (a). for Ax-FPM, (b). for HEAP [22].

Measuring the error introduced by the Ax-FPM and the HEAP for small inputs ranging between 0 and 1, see Figure 15, we notice that, in addition to the smaller magnitude of error when using the HEAP, a different pattern was found: less data dependency and for only 34% of the cases, the HEAP results were higher than the exact multiplier results.

In order to further investigate the impact of different approximate multipliers on the convolution layer output, we produce different heat maps. As shown in Figure 16, we can see that the Ax-FPM is further highlighting the important features by increasing their scores whereas the HEAP is lowering their scores.

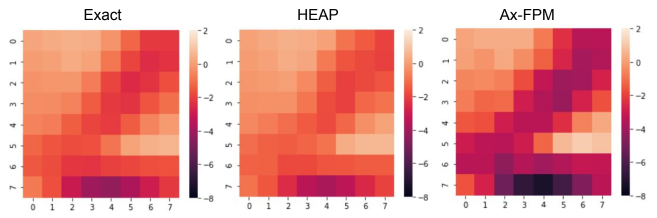

Figure 16: Heat maps of final convolution layer output when using Exact multiplier, Ax-FPM, and HEAP [22].

Energy consumption and Delay We also compare the gain in terms of energy and delay achieved by the two approximate mantissa multipliers. Multipliers are implemented using 45 *nm* technology via the Predictive Technology Model (PTM) using the Keysight Advanced Design System (ADS) simulation platform [27].

Table 9: Energy and delay of different 24×24 approximate multipliers normalized to a conventional 24×24 multiplier.

Multiplier	Average energy	Average delay
Exact multiplier	1	1
HEAP	0.49	0.46
Ax-FPM	0.395	0.235

As shown in Table 9, the AMA5-based mantissa multiplier achieves a considerable gain in performance and energy saving compared to the HEAP. A significant reduction in the delay was also achieved.

Robustness We tried to assess different models transferability, the adversarial examples were generated to fool the exact model. In table 10, we present the percentage of adversarial examples that successfully deceived the approximate models. While the use of the HEAP increased the robustness of the model, the Ax-FPM achieved better results.

Table 10: Attacks transferability success rates for MNIST.

Attack	Exact-based	HEAP-based	Ax-FPM-based
FGSM	100%	16%	12%
PGD	100%	30%	28%
JSMA	100%	42%	9%
C&W	100%	6%	1%
DF	100%	18%	17%
LSA	100%	41%	18%
BA	100%	22%	17%
HSJ	100%	4%	2%

All these results confirm that Ax-FPM is a better choice in terms of robustness and resource and power efficiency.

B. Appendix: Defensive Quantization: Models Architecture

For the fully quantized model, ConvolutionQuant, DenseQuant and reluQuant designate respectively a convolution layer with quantized weights, a dense layer with quantized weights and the relu activation function with its output quantized. Weight Quantized models architecture. ConvolutionQuant and DenseQuant designate respectively a convolution layer with quantized weights and a dense layer with quantized weights.

Table 11: Fully quantized and Weight quantized models architecture.

Fully Quantized	Weight Quantized
ConvolutionQuant	ConvolutionQuant
BatchNorm	BatchNorm
reluQuant	relu
ConvolutionreluQuant	ConvolutionreluQuant
MaxPooling	MaxPooling
BatchNorm	BatchNorm
reluQuant	relu
ConvolutionQuant	ConvolutionQuant
BatchNorm	BatchNorm
reluQuant	relu
ConvolutionQuant	ConvolutionQuant
MaxPooling	MaxPooling
BatchNorm	BatchNorm
reluQuant	relu
ConvolutionQuant	ConvolutionQuant
BatchNorm	BatchNorm
reluQuant	relu
ConvolutionQuant	ConvolutionQuant
MaxPooling	MaxPooling
BatchNorm	BatchNorm
reluQuant	relu
DenseQuant	DenseQuant
BatchNorm	BatchNorm
reluQuant	relu
DenseQuant	DenseQuant
BatchNorm	BatchNorm
reluQuant	relu
Dense	Dense
softmax	softmax

Where ConvolutionQuant, DenseQuant and reluQuant stands for respectively a convolution layer, a dense layer with quantized weights and relu activation function with its output quantized.