DuDAB: A Dual-Staged, Lightweight Detection Methodology for Adversarial Attacks in a Black-Box Setting

Blinded For Review

Abstract

Adversarial attacks pose a critical threat to the reliability of deep neural networks (DNNs), particularly in security-sensitive applications. Traditional detection methods often rely on statistical techniques, such as kernel density estimation, maximum mean discrepancy, or local intrinsic dimensionality, focusing solely on software-based features and neglecting hardware-level metrics. However, such detection strategies are unsuitable for black-box settings. To the best of our knowledge, the only black-box detection mechanism for adversarial attacks is EMShepherd (AsiaCCS 2023).

However, EMShepherd is prohibitive because of complex preprocessing and deployment of Variational Auto-Encoders (VAEs). In this work, we propose a light-weight alternative. Our main contribution is breaking the detection mechanism into two stages: a first stage detector comprising light-weight statistic Root Mean Square (RMS), and a second stage detector comprising a Convolution Neural Network (CNN). The second stage detector is engaged iff the first stage detector fails; this ensures majority of adversarial attacks are detected by statistical tests, rather than complex VAEs. To further the efficacy of our detection method, we experiment upon a production grade SoC (Raspberry Pi 3B) which comprises complex out-of-order multi-threaded cores, concurrent/parallel execution of other processes on the SoC, workload-specific throttling, and several other hardware optimizations lacking in FPGA based accelerators (considered by EMShepherd).

We conduct experiments on standard datasets like MNIST, CIFAR10, and ImageNet on standard models across state-of-the-art adversarial attacks: like FGM, FGSM, L2CarliniWagner, L2PGD, LinfAdditiveUniformNoise, LinfDeepFool, PGD, and LinfPGD. Our results demonstrate a comparable detection accuracy of 87%-98% relative to EMShepherd, while achieving a faster detection time of 50 ms and model size of 1.3 MB. This highlights the robustness, scalability, and practicality of our hardware-centric two-step approach.

CCS Concepts

• **Security and privacy → Systems security**.

Keywords

Adversarial Attack, Hardware Metrics, Side Channel Analysis, Deep Neural Networks

Conference acronym 'XX, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/2018/06
https://doi.org/XXXXXXX.XXXXXXX

ACM Reference Format:
Blinded For Review. 2018. DuDAB: A Dual-Staged, Lightweight Detection Methodology for Adversarial Attacks in a Black-Box Setting. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation email (Conference acronym 'XX)*. ACM, New York, NY, USA, 12 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Adversarial Attacks. Adversarial attacks have become a major challenge in the deployment of deep neural networks (DNNs) across safety-critical applications such as healthcare, autonomous driving, and cybersecurity [31]. These attacks exploit vulnerabilities in DNNs by adding imperceptible perturbations to input data, which can mislead models into making incorrect predictions [11, 46]. Such vulnerabilities pose significant risks, especially in safety-critical applications like autonomous driving and medical diagnostics, where erroneous outputs can lead to severe consequences [2]. Detecting adversarial attacks is therefore crucial to ensure the reliability and security of AI systems.

"White"-Box Detection. Existing detection methods rely heavily on software-based approaches. These include statistical techniques like kernel density estimation [6], maximum mean discrepancy [37], and local intrinsic dimensionality [26], which analyze the distributions of input features, activation patterns, or monitor output distributions [19]. Other popular methods of detecting adversarial attacks include: analysis of intermediate features during execution of adversarial examples [25, 43], and analysis of input distribution [13, 26].

While these methods have shown promise, they come with significant limitations: extensive pre-preprocessing, access to model internals, and large computation resources. As such, these methods are suitable only for *white-box* deployments. However, with the rising need to ensure privacy and security in deployed machine learning applications, it is essential to develop detection methods outside the purview of white-box systems.

A "Black"-Box Solution: In [8], an alternative detection strategy has been proposed in a black-box setting. The main insight of the work is that hardware-level analysis offers a novel and complementary approach for detecting adversarial attacks. Unlike software-based methods, hardware-level characteristics such as electromagnetic (EM) radiation, power consumption, and so provide a direct perspective into the computations performed during model inference. These characteristics are inherently tied to the hardware's operation and are challenging for adversarial attacks to manipulate, making them robust indicators of anomalous behavior. Observing these hardware characteristics has several advantages. First, it enables black-box detection, as the analysis does not require access to the model's architecture, parameters, or input data. Second, hardware-based methods can provide real-time insights into the model's operation without adding significant computational overhead. Lastly, they enhance the robustness of adversarial

detection by leveraging a physical layer that is less accessible to attackers.

Issues with [8]: While [8] takes an important step towards constructing a detection methodology for adversarial attacks in a black-box setting, it has a few shortcomings. First, it requires a non-trivial pre-processing step (i.e. Short-Time Fourier Transform (STFT)) to denoise captured electro-magnetic traces, adding to the turnaround time of detection. Second, it uses Variational Auto-Encoders as the detector, which are both complex and unsuitable for deployment on resource-constrained devices, as well as heavily influence the detection turnaround time.

Given this context, in this work, we ask the following question:

*Can a lightweight, real-time adversarial detection method be developed in a **black-box** setting, while eliminating the need for complex pre-processing and achieving acceptable detection accuracy for complex platforms like out-of-order SoCs?.*

In this work, we propose a lightweight, real-time, generic method for adversarial detection that leverages hardware characteristics, specifically EM radiations, eliminating the need for complex pre-processing. Our approach is based on the intuition that *adversarial inputs and benign inputs activate different neurons, leading to distinct computational patterns* [15], which manifests itself as distinguishable EM emissions. We provide an intuition of our solution.

First, when considering the lightweight nature of the solution, we note that [8] is prohibitive in the sense that it uses complex pre-processing Short-Time Fourier Transform (STFT) to denoise the EM traces, followed by an inference by Variational Auto-Encoders to infer adversarial/benign execution. This is because of the need to demarcate denoising and inference. Instead of this, we propose to use a lightweight statistic- Root Mean Square (RMS) of the captured Electromagnetic traces for detection. Essentially, we demonstrate that for adversarial attacks, RMS exhibits inherent denoising which is sufficient for single-shot detection of adversarial attacks, implying the need to perform complicated pre-processing. This switch from Variation Auto-Encoders to a lightweight statistic (i.e. RMS) is at the core of building our lightweight detector.

As we later also demonstrate, while RMS is sufficient for detection of *most* attacks, it offers average detection accuracy rates for *some* attacks (cf. Sec. 4). As such, we couple our detector with a *second* layer of detection: a lightweight Convolution Neural Network (CNN). Thus, for each attack, we first use RMS to filter out attacks that are easily detectable, and only fall back on CNNs for complex attacks. Unlike [8] therefore, we do not rely on deep learning for detection of *every* attack; we rather use CNNs where-ever necessary, thereby ensuring the relative light-weight nature of our solution wrt. [8].

Second, to demonstrate the efficacy of our solution on complex platforms, we choose our victim device as Raspberry Pi 3B (instead of FPGA accelerators tested in [8]). Testing upon production level SoCs allows us to factor in chip-level effects like dynamic voltage frequency scaling, noise from other processes due to multi-core SoC, factors like kernel scheduling and so on. This allows us to test our methodology in realistic deployment scenarios (instead of FPGA accelerators where models run usually in isolation). Moreover, since our focal point is development of a detector in a **black-box** setting, choice of Raspberry Pi 3B allows our solution to be also deployed on ARM TrustZone[1] which is widely proposed for privacy-preserving machine learning deployments [17, 24, 30, 45].

Unlike EMShepherd, our method eliminates the need for pre-processing or feature extraction, making it simpler and more practical for real-world applications. By focusing on RMS values, our approach offers a hardware-centric, model-agnostic solution for detecting adversarial activity. It is particularly suited for edge devices, where computational resources are limited, and model architectures are diverse. Our contributions can be listed as follows:

- We propose a lightweight, dual-staged detection methodology for detection of adversarial attacks in a black-box setting. At the first stage, adversarial attacks are evaluated by a lightweight statistic- RMS. We fall back on a second stage CNN when the first stage fails to reliably detect the attack. This allows our solution to have much better detection turnaround time compared to the Variational Auto-Encoders [8]. Moreover, our detection approach is class agnostic, unlike the approach in [8] which is class specific.
- We validate our detection methodology on a production level SoC (Raspberry Pi 3 platform) using diverse datasets, including MNIST, CIFAR10 and ImageNet, and evaluate its effectiveness across various DNN models and adversarial attacks. Using the proposed methodology, we are able to achieve an accuracy close to $92-98\%$ for attacks like FGM, FGSM, L2CarliniWagner, L2PGD, PGD and an accuracy close to $89-90\%$ for attacks like LinfPGD, LinfAdditiveUniformNoise, LinfDeepFool.
- Finally, we compare our detection methods with existing methods (both white-box and black-box) of detecting adversarial attacks. The results show that our detection methodology achieves comparable detection accuracy, while considerably improving the detection turnaround time and model complexity. The RMS detection only requires 0.46 ms and the CNN model detection requires 50 ms. The model size is approximately 1.3 MB making it suitable for lightweight applications.

The rest of the paper is structured as follows: In Section 2, we provide a brief overview of adversarial attacks and existing methods for detecting such attacks. Section 3 briefly illustrates the proposed methodology. In Section 4, we present our experimental results followed by a conclusion in Section 5.

2 Background

This section briefly overviews the adversarial attacks and the existing methods to detect such attacks.

2.1 Adversarial Attacks

Adversarial attacks exploit the vulnerabilities of deep neural networks (DNNs) by introducing subtle, often imperceptible perturbations to input data, causing models to make incorrect predictions. These attacks have significant implications for critical applications, such as autonomous vehicles, healthcare, and financial systems, where erroneous decisions can lead to severe consequences. Adversarial attacks are particularly problematic because they can bypass

[1]The Trusted Execution Environment for ARMv8

traditional accuracy metrics and are difficult to detect without specialized techniques [3].

The types of adversarial attacks vary in sophistication and objectives. Gradient-based attacks, such as Projected Gradient Descent (PGD) [27] and Fast Gradient Sign Method (FGSM) [11], are among the most studied and effective methods. These attacks leverage gradient information from the model's loss function to craft perturbations that mislead the network. Targeted attacks aim to force the model to output a specific incorrect label, while non-targeted attacks simply cause the model to misclassify the input. DeepFool is another adversarial attack proposed in [34] which introduces an efficient method to compute the minimal perturbations necessary to cause a classifier to misclassify an image. The Additive Uniform Noise (AUN) [36] adversarial attack probes the robustness of a model by introducing independent and identically distributed (i.i.d.) uniform noise into the input. The Carlini Wagner(CW) attack [2] approaches adversarial example generation as an optimization problem, aiming to identify the minimal perturbation required to mislead the target model into making an incorrect classification.

Adversarial examples pose a unique challenge because they often appear visually indistinguishable from benign inputs. This makes them particularly insidious in real-world settings, as humans cannot reliably detect them. Moreover, adversarial attacks exploit the high-dimensional nature of DNNs, where small changes in input space can cause significant changes in output space due to the model's non-linear decision boundaries. Adversarial inputs can activate neurons in patterns that differ significantly from those activated by benign inputs, leading to computational footprints that are distinct and measurable [9].

2.2 Detecting Adversarial Attacks

Several methodologies have been proposed for detecting adversarial attacks, falling into categories such as statistical analysis, input transformations, and prediction inconsistencies [43]. Statistical techniques, such as kernel density estimation or maximum mean discrepancy, rely on calculating the distance between input representations and known distributions. While effective in specific scenarios, these methods often require large datasets for training and fail to detect individual adversarial instances, limiting their practical applicability. Input transformation approaches, like applying feature-squeezing techniques or using autoencoders, aim to filter adversarial perturbations by altering the input representation [29]. These methods, while computationally lightweight, often degrade the accuracy of benign inputs or are evaded by adaptive attacks that exploit the transformation process. Another class of approaches involves measuring prediction inconsistencies across multiple model evaluations, such as using Bayesian neural networks with dropout [10]. Although promising, these methods depend on ensemble-like predictions, which increase computational costs and may not scale efficiently for real-time applications. Overall, the limitations of these methods highlight the need for robust, lightweight, and generalizable adversarial detection frameworks that do not overly depend on complex preprocessing or dataset-specific optimizations.

2.3 EMShepherd

Hardware characteristics, such as electromagnetic (EM) radiation, power consumption, and execution timing, provide a unique and robust perspective for detecting adversarial activity in DNNs. These metrics are directly tied to the physical computations performed by the hardware, offering insights that are difficult for adversarial inputs to manipulate. Unlike software-based detection methods, which rely heavily on analyzing input distributions or model-specific features, hardware-level analysis observes the underlying behavior of the system itself, making it inherently model-agnostic [33].

One of the key advantages of using hardware-level characteristics is their applicability in black-box settings, where the model architecture and parameters are inaccessible [20]. By monitoring physical signals during inference, it becomes possible to detect anomalies caused by adversarial inputs without requiring access to the model's internal gradients or activations. This makes hardware-based methods particularly valuable in scenarios where intellectual property or privacy concerns limit access to the model.

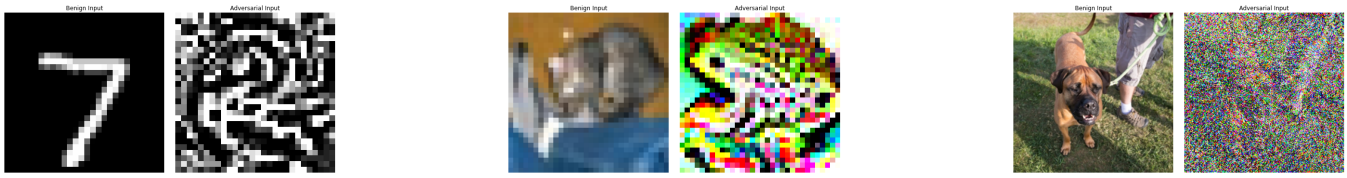
Additionally, hardware-level characteristics offer increased tamper resistance. While adversarial attacks are designed to deceive the model's decision-making process, they cannot directly influence the physical signals generated during computation. For instance, adversarial inputs that alter neuron activation patterns in gradient-based attacks, such as PGD, lead to distinct changes in electromagnetic emissions or power profiles. These changes are measurable and provide a reliable indicator of adversarial activity [8]. EMShepherd [8] is a framework designed to detect adversarial samples using side-channel electromagnetic (EM) leakage. It operates in a black-box scenario, requiring no prior knowledge of the model's parameters, structure, or inputs. By collecting EM traces during a model's execution, the system processes these signals and trains EM classifiers and anomaly detectors to identify adversarial patterns. The EM emanations reveal discrepancies in computations when adversarial examples are processed compared to benign inputs. However, EMShepherd faces significant challenges. Its preprocessing pipeline, including Short-Time Fourier Transform (STFT) for denoising and generating spectrograms for class specific feature extractions, which makes it computationally intensive and introduces a substantial overhead. The requirement to partition traces into multiple segments and train separate EM classifiers for each segment adds further complexity and makes it less practical for real-time or lighweight applications.

3 Proposed Methodology

In this section, we explain the intuition and design our detection methodology. To do this, we first establish the main properties of adversarial attacks and empirically argue how they are connected with electromagnetic emissions. We then establish how our chosen statistic- RMS exhibits inherent denoising and can be converted into a lightweight detection methodology, which we subsequently elaborate.

3.1 EM Emissions from Adversarial Attacks in Production-Level SoCs

Before we describe our detection method, we first establish the nature and cause of EM emissions under adversarial attacks on SoCs.



(a) MNIST (b) CIFAR10 (c) ImageNet

Figure 1: Differentiating Adversarial and Benign Inputs for different datasets

In [8], the detection of adversarial attacks is attributed to properties of the FPGA accelerators. On SoCs (which are the focus of this work) however, differential EM emissions for adversarial attacks have a slightly different root cause: Dynamic Voltage Frequency Scaling (DVFS).

Modern SoCs (like the one we consider in this work) have dedicated DVFS to balance performance and energy consumption in the core package [12, 21, 38, 41]. Essentially, DVFS relies on workload characteristics to choose a power budget for every core, that ends up throttling the core to a certain frequency. This ensures peak performance but only when absolutely required, while also conserving energy whenever possible. From the perspective of an observation channel, DVFS induced throttling of a CPU core is an interesting tool as it gives a **black-box** insight into the workload characteristics. In other words, DVFS induced CPU throttle can allow differentiating between workloads without the knowledge of their internals. This holds as long as the workloads differ in their execution context, which essentially requires the CPU to throttle them differently. Finally, DVFS induced CPU throttles are manifested in several forms:

(1) Varying power consumption [22]
(2) Varying CPU frequency [23, 39, 40, 44]
(3) Varying CPU temperatures [32]
(4) Varying Electromagnetic Emissions [4, 5]

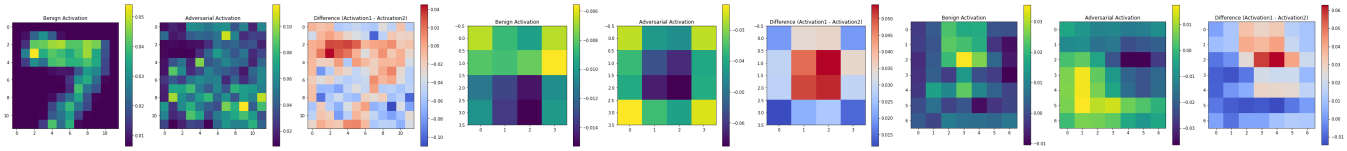
Out of these four options, we consider Electromagnetic Emissions as the measurement channel in this work for a number of reasons. First, all other approaches require software intrusion (i.e. software access to the interfaces which readout these telemetric data-points), which would be prohibited in deployments using a Trusted Execution Environment like ARM TrustZone. On the other hand, equipment measuring Electromagnetic Emissions is completely non-intrusive, and requires no modifications whatsoever to on-device code (including the code executing within ARM TrustZone). Second, on-device telemetric interfaces may have software-induced restrictions in terms of resolution [32] or obfuscation [16], whereas measurements related to Electromagnetic Emissions have no such restrictions because the measurement apparatus is external to the device. Finally, to the best of our knowledge, the only black-box detection method is proposed by [8]; by using Electromagnetic emissions in our detector, we are able to perform a fair evaluation with state-of-the-art.

We now design experiments to empirically verify that adversarial inputs often activate different regions of the network compared to benign inputs, leading to distinct activation patterns at various layers of the DNN [2], and hence induces different DVFS induced CPU throttles, affecting EM emissions. Without loss of generality, we consider `LinfPGD` for our investigation. Consider Fig. 1 which plots benign/adversarial examples for the `LinfPGD` attack on different datasets. Note particularly how the benign inputs and adversarial inputs come from significantly different distributions. Fig. 2 captures the differential neuron activation maps for different datasets respectively. Divergence in such neuron activation map stems from the adversarial perturbations crafted to maximize the loss function, causing the network to deviate from its standard decision boundary and engage irrelevant or misleading features [11].

These activation pattern differences upon processing of adversarial inputs alter the computational paths, memory accesses, and arithmetic operations. These *differential* executions between benign and adversarial examples are manifested as differential workload characteristics for the DVFS mechanism, which allots distinguishable CPU throttle for benign/adversarial inputs. Also note that EM emissions are closely tied to the underlying computations performed by the hardware under varying DVFS induced CPU throttle (cf. Fig. 3, where the EM emissions from benign and adversarial inputs are statistically distinguishable). Thereby, these differences can be statistically quantified using appropriate metrics. By leveraging such hardware variations, adversarial attacks can potentially be detected in real-time and in the **black-box** setting (without modifying the model or requiring access to its parameters).

Insight: On out-of-order, multi-core SoCs, complex mechanisms like DVFS operate to trade-off performance and energy consumption. DVFS takes into account *workload characteristics* (an amalgamation of several factors like workload instruction stream, memory access patterns, peripherical communications, and so on) to decide a throttle to the CPU.
For the *same* workload (i.e. victim model), adversarial attacks differ from benign execution in terms of input data distributions, which causes differential neural activations and thereby different *workload characteristics*. DVFS thereby throttles a CPU differently under adversarial execution than it does under benign execution. Such differential CPU throttle then influences telemetric readouts like power, temperature, CPU frequency, and EM Emissions.

As aforementioned, to keep our detection methodology non-intrusive as well as to perform a fair comparative evaluation with [8], in subsequent sections, we only consider Electromagnetic Emissions as measurements of our tool. It is an interesting future problem to rehash our methodology using some of the other telemetric measurement modalities like power, temperature, and CPU frequency.



(a) MNIST (b) CIFAR10 (c) ImageNet

Figure 2: Differential Neuron Activation Maps for Adversarial/Benign Inputs for different datasets

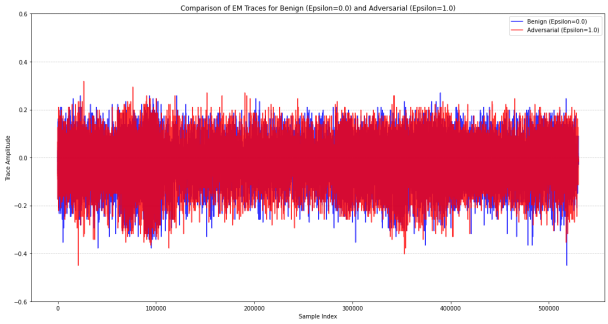


Figure 3: Comparison of Trace for $\epsilon = 0.0$ representing benign input and $\epsilon = 1.0$ representing adversarial input

3.2 Denoising and Attack Detection Using RMS

Having established the link between adversarial attacks and EM emissions (through DVFS induced CPU throttle), we now discuss our rationale for the choice of statistical tool in our detection methodology. In [8], raw EM traces are operated upon by Short-Time Fourier Transform (STFT) [35] to denoise them. STFT [35] is a widely used method for analyzing the frequency content of non-stationary signals over time. It involves segmenting the signal into overlapping windows and applying the Fourier transform to each segment, resulting in a time-frequency representation. STFT is effective for denoising, especially when noise characteristics vary across frequencies. However, from the perspective of our use case of developing a *lightweight* detection methodology, STFT has certain limitations. First, STFT requires multiple Fourier transforms, making it computationally intensive, especially for long signals or real-time applications, Second, there is an inherent trade-off between time and frequency resolution in STFT; improving one leads to a compromise in the other.

In our work, we choose RMS over the captured EM emissions as the first level of detection. In signal processing, RMS is a statistical measure of the magnitude of a varying signal. While RMS is primarily used to quantify the energy or power of a signal, it can also play a role in denoising [28]. By calculating the RMS value over different segments of a signal, one can identify portions with higher variations. This information can used as a checkpoint for identifying any deviations from ideal computation [18]. Moreover, using RMS-based for denoising is more computationally efficient since: • Calculating RMS is straightforward and less computationally demanding compared to performing multiple Fourier transforms. • Due to its simplicity, RMS calculation can be implemented in real-time systems with limited processing power.[2].

We now test the ability of RMS to segregate adversarial attacks from benign executions. Without loss of generality, we consider `LinfPGD` for our investigation. Across MNIST and CIFAR10, we performed `LinfPGD` attack considering several different epsilons, and collected the EM emission traces. Note that higher epsilons imply higher probability of successful adversarial attacks. We then considered a fixed window size of 10000 and computed RMS of each window. As shown in Fig. 4, the primary Y-axis (left) plots the trace amplitudes for different epsilons of `LinfPGD` attack[3]. The superimposed secondary Y-axis (right) plots the RMS deviations in the logarithmic scale.

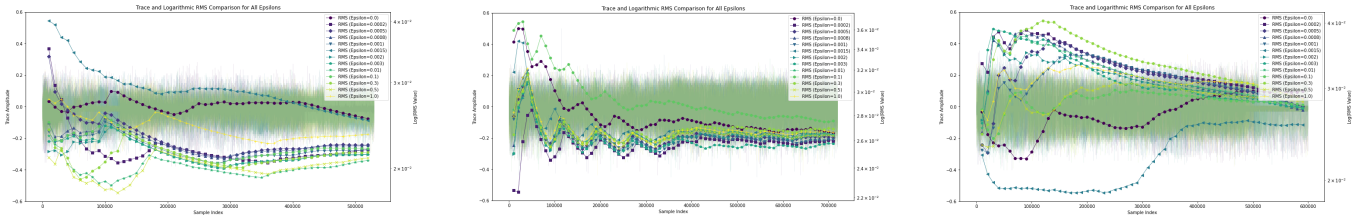
We observe that as the observation interval (i.e. the X-axis) increases, RMS trends for the different attack epsilons plateau, giving us fairly consistent values[4]. RMS thus allows us to develop templates, and also offers an opportunity to build a low-cost adversarial attack detector.

Insight: RMS offers a lightweight alternative to STFT for denoising EM traces because of its tendency to average over EM emissions over a period of time. Furthermore, unlike STFT, RMS does not need a separate attack detector since detection is inherent in RMS. Consider `LinfPGD` on MNIST, and consider the set $R = \{r_{0.0}, r_{0.0002}, r_{0.0005}, \cdots, r_{0.5}, r_{1.0}\}$ to denote the set of RMS templates generated in the **offline** phase (cf. Fig. 4).
During the **online** phase, detection thereby involves capturing EM emission from a *single* execution of victim model, computing the attack RMS a, and computing the pair-wise statistical distance of a from each element in R. We say an attack is being performed if $| a - r_i | \leq \lambda$, for some preset threshold λ and where i represents an epsilon that leads to appreciable drop in model accuracy (i.e. a successful adversarial attack).

3.3 End-to-End Detection Methodology

We now explain our final end-to-end detection methodology as shown in Fig. 5. In Sec. 3.2, we introduced RMS as the statistical tool of choice having lightweight denoising and detection capabilities. In order to alleviate any false positives from RMS, we append a *second-stage* lightweight CNN to correctly classify cases which RMS mis-classifies.

[2]We refer to [1] for a comparative analysis of several denoising techniques in the context of speech signals, highlighting the trade-offs between computational complexity and denoising performance.
[3]We observed similar trends for other attack methods as well.
[4]Note that the RMS values in the plots are in logarithmic scale. Slight differences hence are sufficient for differentiation.



(a) LinfPGD Attack on MNIST Dataset (b) LinfPGD Attack on CIFAR10 Dataset (c) LinfPGD Attack on ImageNet Dataset

Figure 4: Comparison of trace amplitudes and cumulative RMS values during LinfPGD attack for various epsilon values during DNN inference.

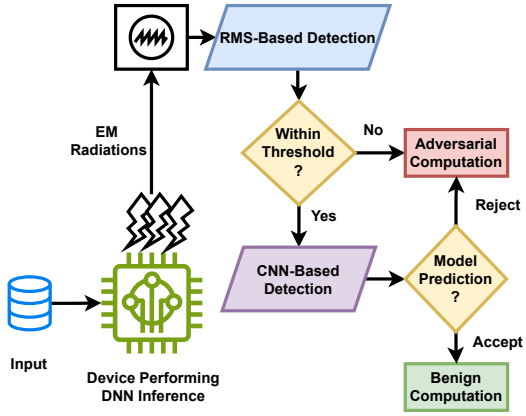


Figure 5: Dual-Staged Detection Methodology for Adversarial Attack

Offline Phase. In this phase, we create templates for our detectors. This involves running the attacks-under-test on the target platform against the victim models, and collecting EM emissions. We then perform the following steps:

(1) **Stage 1: RMS Detector**. As detailed in Sec. 3.2 takeaway, we construct RMS template against different epsilons for each attack, and label each template as benign/adversarial. We denote such templates as $R^{\epsilon}_{\mathcal{A}}$ where $0 \leq \epsilon \leq 1$ denotes the attack epsilon, and $\mathcal{A}$ denotes the attack type from FGM, FGSM, L2CarliniWagner, L2PGD, LinfAdditiveUniformNoise, LinfDeepFool, PGD, and LinfPGD. We also note the standard deviation $\sigma(R^{\epsilon}_{\mathcal{A}})$ for the template.
(2) **Stage 2: CNN Detector**. This involves training the CNN with the captured EM Emissions and their corresponding benign/adversarial label. Our training involves establishing a class-agnostic baseline: train the ML model to establish a baseline independent of prediction classes, focusing on identifying patterns in EM traces indicative of adversarial behavior[5]. We denote such models as a set of binary classifiers $\mathcal{M}_{\mathcal{A}}$ where $\mathcal{A}$ denotes the attack type from FGM, FGSM, L2CarliniWagner, L2PGD, LinfAdditiveUniformNoise, LinfDeepFool, PGD, and LinfPGD.

[5]We note that in [8], the Variational Auto-Encoders are trained in a class-specific manner, while our approach is class-agnostic.

Online Phase. In the online phase, we perform the following sequence of steps:

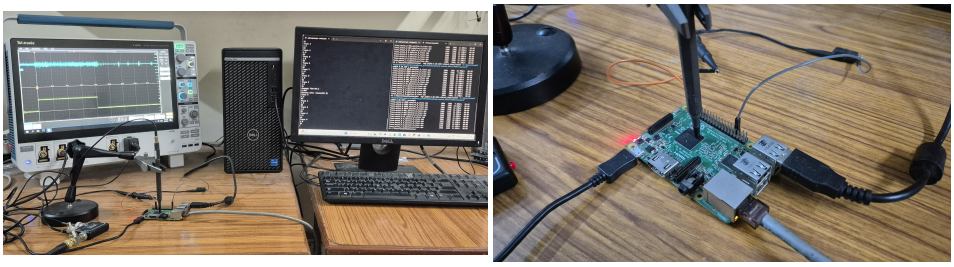
(1) **Data Acquisition**: Collect EM emission trace T during the vicim model inference in **a single shot**. In other words, our detection methodology does not require multiple re-runs of the victim model.
(2) **RMS based Detection**: To detect whether an attack $\mathcal{A}$ is operational, compute RMS of T (say R^T), and compute pairwise statistical distance $\mid R^T - R^{\epsilon}_{\mathcal{A}} \mid$ for all ϵ existing in template $R^{\epsilon}_{\mathcal{A}}$. If $\mid R^T - R^{\epsilon}_{\mathcal{A}} \mid \leq \sigma(R^{\epsilon}_{\mathcal{A}})$ (i.e. RMS of T is within the standard deviation range of the templated RMS) **and** ϵ leads to appreciable accuracy drop in victim model, we state that the attack $\mathcal{A}$ is ongoing.
(3) **Fallback Mechanism**: In case the previous stage classifies $\mathcal{A}$ as benign, we engage the binary classifier $\mathcal{M}_{\mathcal{A}}$ on T to ascertain whether an attack is operational or not.

By combining these two steps, our method ensures robustness and scalability, leveraging the simplicity of RMS-based detection and the adaptability of ML-based models.

3.4 Comparison with EMShepherd

In this section, we provide a comparative analysis with EMShepherd, as to the best of our knowledge, it is the only detection framework that operates in the same black-box setting as us. We note the following points of comparison of our methodology with EMShepherd that allows us to achieve tighter detection turnaround times while retaining comparable detection accuracy:

(1) **Setting**: Both EMShepherd and our detection methodology operate in a **completely black-box setting**. Nowhere do we use any information about the model internals, intermediate neural activation maps, and other telemetric information for detection.
(2) **Target Device**: While EMShepherd operates upon FPGA accelerators, our methodology is experimented upon a production level out-of-order SoC. This difference has a few implications: Our **leakage source** for EM emissions (i.e. DVFS induced throttle) is **more complicated** than EMShepherd's (i.e. hardware), and our methodology can also be **extended trivially to ARM TrustZone** in the use-cases of privacy-preserving machine learning.



(a) Host PC communicating with device and oscilloscope

(b) Raspberry Pi 3 under test

Figure 6: Setup for acquisition of EM Traces

(3) **Denoising Methodology**: EMShepherd relies upon Short-Time Fourier Transform (STFT) for denoising, which involves multiple Fourier Transform computations over the collected traces. On the other hand, **we rely on a lightweight statistic: RMS, to achieve acceptable level of denoising**. Furthermore, STFT requires a separate detector in order to classify an execution as benign/adversarial. However, as detailed in Sec. 3.2, **RMS has inherent detectability by simple analysis of pair-wise statistical distance**. Thereby, the choice of moving from STFT to RMS allows us to improve upon detection turnaround time.

(4) **Reliance on Machine Learning detectors**: EMShepherd relies on complex Variational Auto Encoders for detection. On the other hand, we rely on a dual-staged detection methodology. The first stage is inherently kept completely reliant on statistical properties. Thus, **our detection methodology does not rely on machine learning for *all* attacks**. In case of false positives, a fallback mechanism is initiated that relies on a CNN for a second opinion. As Sec. 4.4 elaborates, the architecture of our CNN is extremely lean and simple. Thereby, we ensure that for cases where we do need to rely on machine learning, **our CNN detector is still more lightweight than EMShepherd's Variational Auto Encoders**.

4 Experimental Results

In this section, we provide empirical results of our detection methodology on different public datasets like MNIST, CIFAR10 and ImageNet.

4.1 Datasets and Models

Datasets. For our experiments, we have utilized public datasets MNIST, CIFAR10 and ImageNet, each offering unique challenges and insights for our analysis. MNIST is a benchmark dataset containing 70,000 grayscale images of handwritten digits (0-9) with a resolution of 28×28 pixels. It is widely used for testing classification algorithms due to its simplicity and suitability for entry-level deep-learning tasks. CIFAR10, on the other hand, comprises 60,000 color images categorized into 10 distinct classes, such as airplanes, cars, and animals. With its 32×32 resolution and diverse content, CIFAR10 serves as an excellent dataset for testing models on small-scale image classification tasks. The significant variations in size, complexity, and scale among these datasets enabled us to assess the performance and generalizability of our methodology across diverse scenarios. We are using Foolbox [36], a Python library that helps to easily run adversarial attacks against machine learning models like DNN for our experiments. Lastly, ImageNet is a large-scale dataset containing over 14 million images across thousands of classes, providing a robust platform for training and evaluating models on complex visual recognition tasks.

Victim Model. For performing inference on the MNIST dataset, we used a Foolbox zoo model with two convolutional layers (32 and 64 filters), ReLU activations, max-pooling, dropout, flattening, and two fully connected layers (128 and 10 units) for multi-class classification [36]. Furthermore, we utilized the Wong2020FastNet [42] model to perform inference on the CIFAR10 dataset, leveraging its robust architecture and adversarial training capabilities to evaluate classification performance effectively. For inference on the ImageNet dataset, we used the MobileNet V2 [14] model, which is designed to be computationally efficient and is well-suited for lightweight devices and low-resolution images.

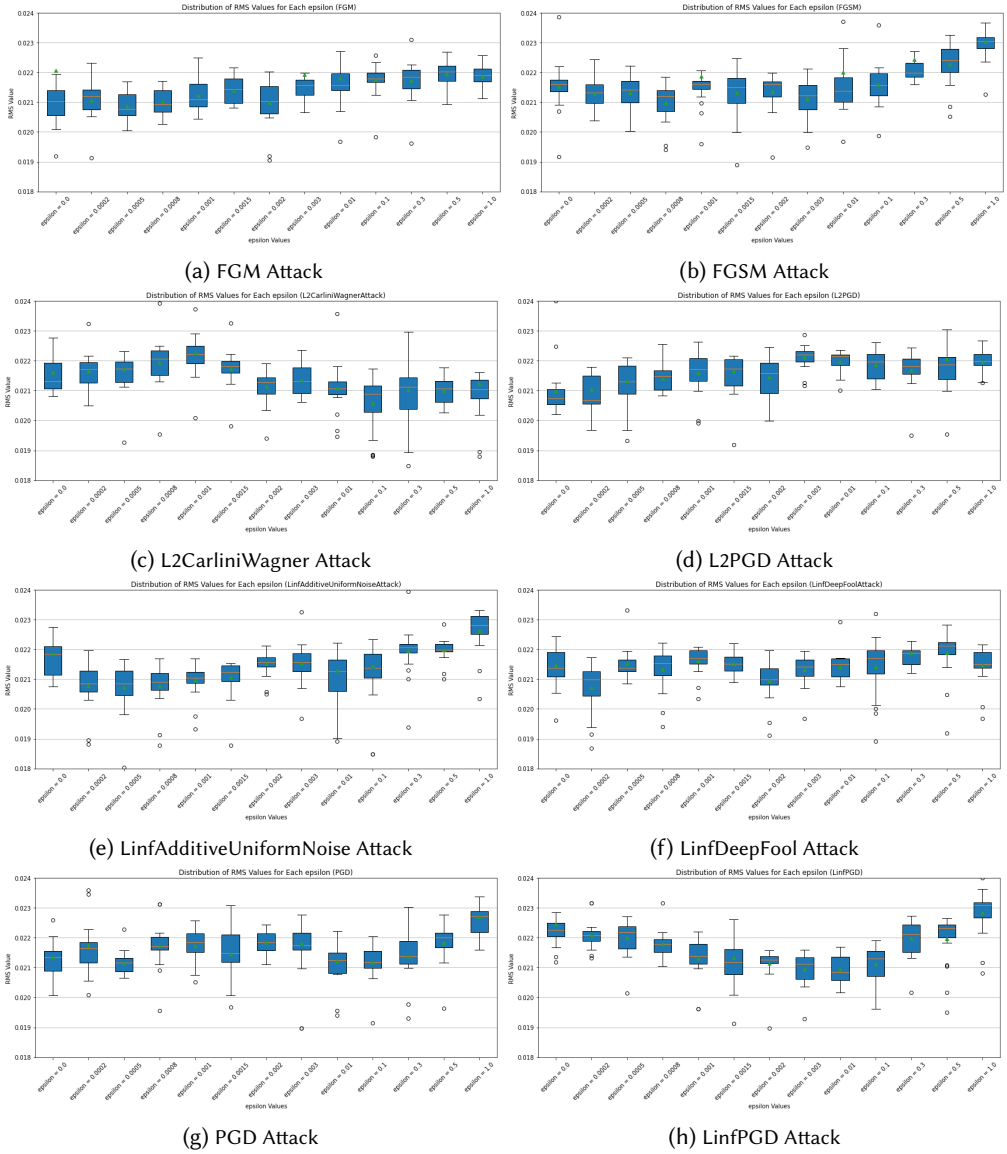
4.2 Hardware Platform

Our experiments have been performed on a Raspberry Pi 3 platform which is a compact and versatile System-on-Chip (SoC) designed for a wide range of computing applications. It features a Quad-Core 1.2GHz with 64-bit CPU, which employs an out-of-order execution architecture to enhance performance by dynamically reordering instructions for efficient execution. With 1GB of RAM, the Raspberry Pi 3 is well-suited for lightweight multitasking and running various operating systems, including the official Raspbian OS. Fig. 6 illustrates our side-channel analysis setup to observe the EM traces where we use a Tektronix MSO64B mixed signal oscilloscope equipped with RF near-field probes to acquire EM power traces from the target board. For MSO was controlled by the device using the VISA instrument access protocol.

4.3 RMS-Based Detection Results

We first discuss empirical results for our stage 1 detection using RMS (on MNIST, CIFAR10 and ImageNet datasets).

MNIST Dataset: Fig. 7 illustrates our observations of RMS values from EM traces under various adversarial attack settings on the MNIST dataset. For each attack, 20 observations were collected across different epsilon values, with those corresponding to no accuracy drop in the victim model considered as benign input executions. The RMS values were subsequently analyzed to identify



(a) FGM Attack (b) FGSM Attack

(c) L2CarliniWagner Attack (d) L2PGD Attack

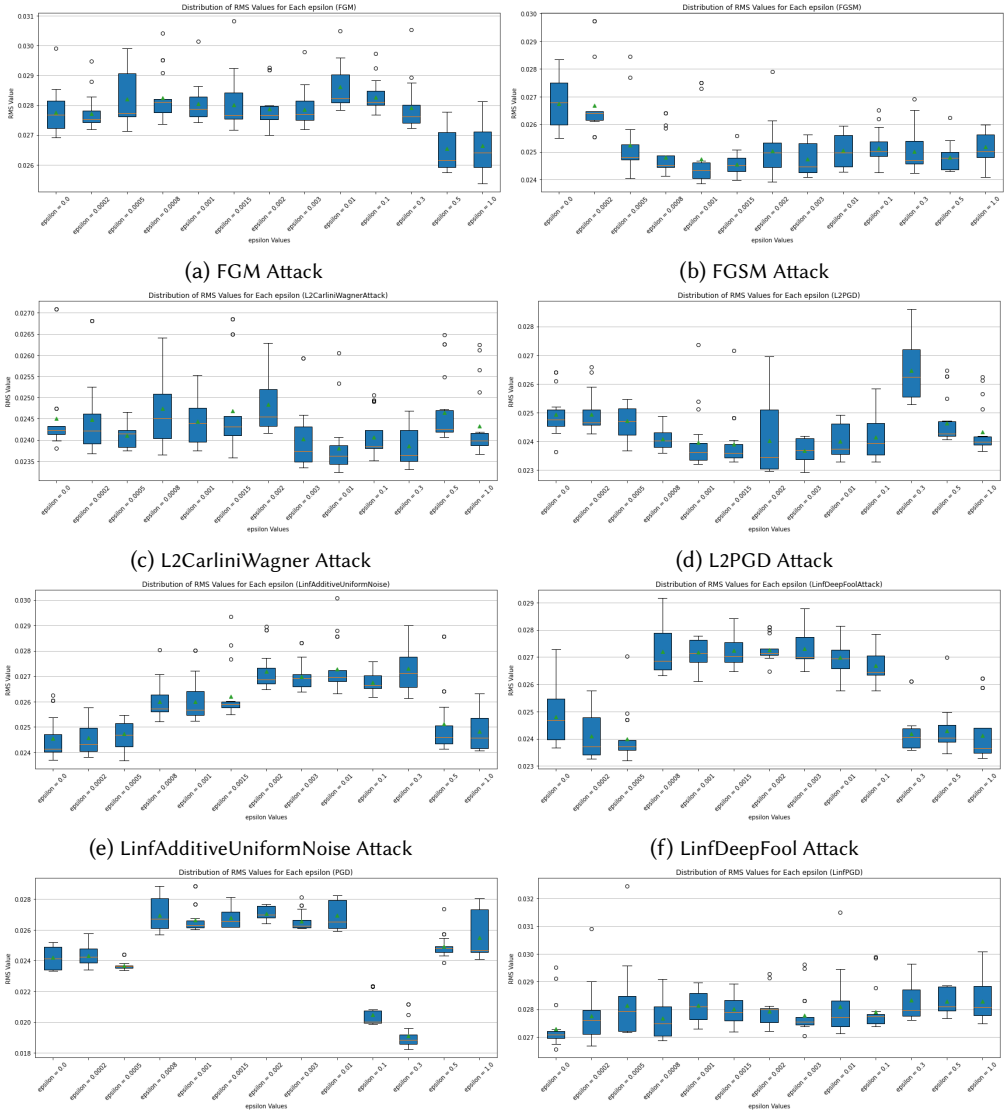
(e) LinfAdditiveUniformNoise Attack (f) LinfDeepFool Attack

(g) PGD Attack (h) LinfPGD Attack

Figure 7: Boxplots showing the distribution of RMS values for each epsilon value across various adversarial attacks on the MNIST dataset. The prediction accuracy drops at: (1) For attacks such as FGM, L2PGD, and L2CarliniWagnerAttack, the accuracy begins to drop only when epsilon reaches 1.0. (2) For LinfAdditiveUniformNoiseAttack, the accuracy drop starts at epsilon = 0.5. (3) For FGSM, LinfDeepFoolAttack, LinfPGD, and PGD, the accuracy drops significantly after epsilon = 0.1.

patterns indicative of adversarial inputs. **A general trend we observe is that for all attacks, it is evident that as the epsilon value increases, there is a deviation in RMS values compared to benign inputs**. This deviation is aligned with the point where the model's accuracy begins to drop, providing a potential threshold for adversarial detection. If the RMS value deviates beyond a safe, predefined range, we raise a flag to indicate the presence of adversarial inputs during inference.

For instance, consider the LinfAdditiveUniformNoise attack depicted in Fig. 7e. At epsilon = 0.0, which corresponds to benign input, the RMS values are observed to range between 0.021 and 0.022. As epsilon increases to 0.5, a noticeable drop in accuracy occurs, and the RMS values exceed 0.022. This clearly signals adversarial behavior, enabling the detection mechanism to raise a flag for such inputs. Similarly, for the FGM, FGSM, and L2CarliniWagner attacks, the graphs indicate that for smaller epsilon values (e.g., 0.0 to 0.001), the RMS values are tightly clustered, reflecting benign-like behavior. However, as epsilon increases to 0.5 or 1.0, the RMS values exhibit significant deviations, indicative of adversarial activity. For the LinfPGD and PGD attacks, the RMS values remain tightly distributed for smaller epsilon values (e.g., 0.0 to 0.0008), consistent with benign behavior. However, as epsilon reaches 0.1, where the



(a) FGM Attack (b) FGSM Attack

(c) L2CarliniWagner Attack (d) L2PGD Attack

(e) LinfAdditiveUniformNoise Attack (f) LinfDeepFool Attack

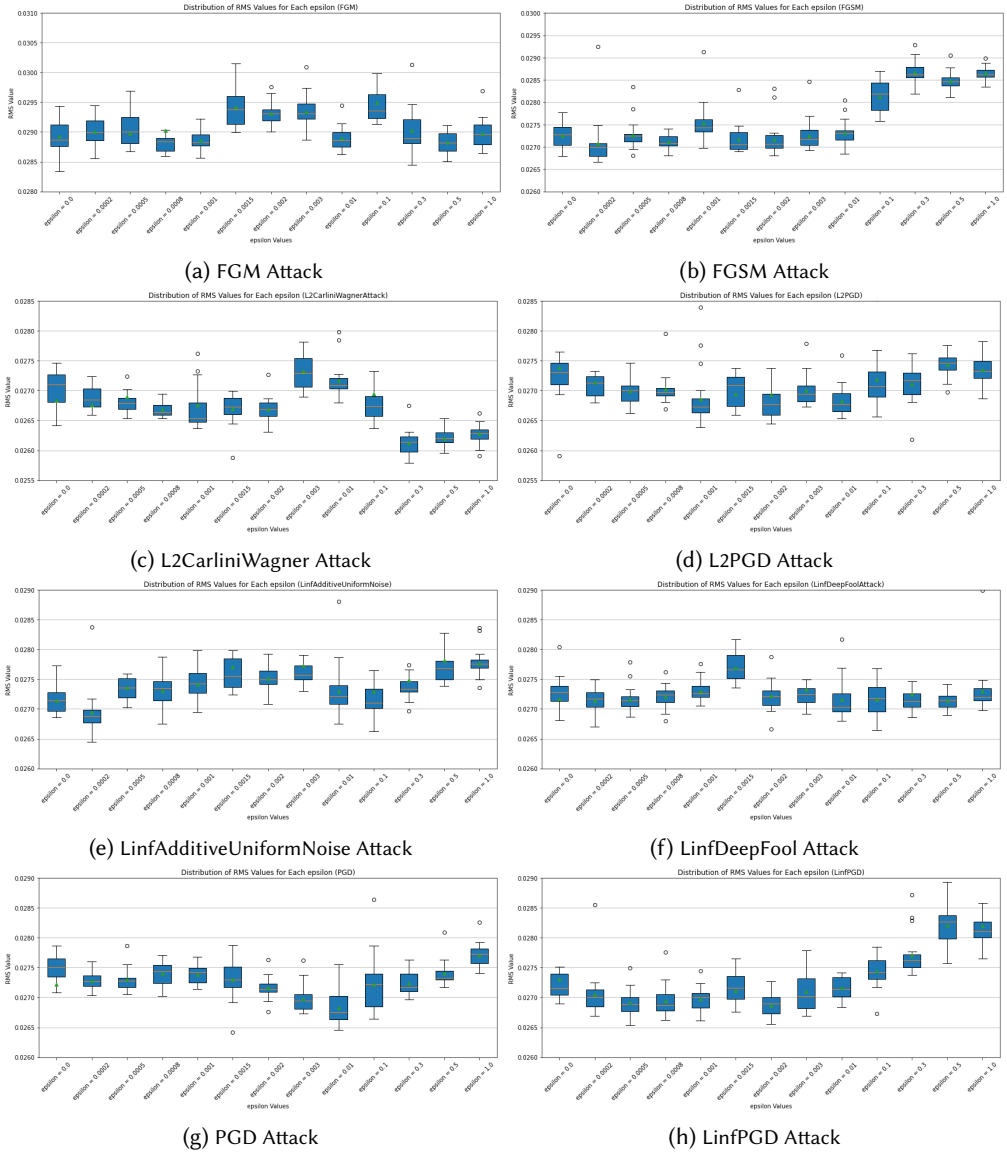
(g) PGD Attack (h) LinfPGD Attack

Figure 8: Boxplots showing the distribution of RMS values for each epsilon value across various adversarial attacks on the CIFAR10 dataset. The prediction accuracy drops at: (1) For attacks such as FGM, L2PGD, and L2CarliniWagnerAttack, the accuracy begins to drop only when epsilon reaches 0.1 (2) For LinfAdditiveUniformNoiseAttack, the accuracy drop starts at epsilon = 0.3 (3) For FGSM, LinfDeepFoolAttack, LinfPGD, and PGD, the accuracy drops significantly after epsilon = 0.0015.

model begins mispredicting, the RMS values deviate significantly, signaling adversarial activity.

CIFAR10 Dataset: Fig. 8 shows the impact of various adversarial attacks on the RMS values of EM traces under different epsilon values for CIFAR10 dataset. Similar to the previous case, for each epsilon value, 20 observations were collected to analyze the RMS values under various adversarial attack settings. For instance, in the case of the PGD attack, as shown in Fig. 8g, the RMS values for benign inputs (epsilon = 0.0) lie within a stable range, indicating normal behavior. However, as the epsilon value increases beyond 0.0005, the RMS values deviate significantly, surpassing the benign threshold. This deviation aligns with a noticeable accuracy drop in the model, thereby signaling adversarial activity.

Similarly, for attacks like FGSM, and L2CarliniWagner, the RMS values for smaller epsilon values (e.g., 0.0 to 0.001) remain tightly distributed, reflecting benign-like behavior. However, as epsilon increases (e.g., to 0.1 or higher), the RMS values exhibit substantial variations, indicating adversarial interference. For attacks such as LinfPGD and L2PGD, benign inputs at smaller epsilon values (e.g., 0.0 to 0.0008) have consistent RMS values, but with higher epsilon values, the RMS deviations become pronounced, corresponding to



(a) FGM Attack (b) FGSM Attack

(c) L2CarliniWagner Attack (d) L2PGD Attack

(e) LinfAdditiveUniformNoise Attack (f) LinfDeepFool Attack

(g) PGD Attack (h) LinfPGD Attack

Figure 9: Boxplots showing the distribution of RMS values for each epsilon value across various adversarial attacks on the ImageNet dataset. The prediction accuracy drops at: (1) For attacks such as FGM, L2PGD, LinfAdditiveUniformNoise and L2CarliniWagnerAttack, the accuracy begins to drop only when epsilon reaches 0.3. (2) For FGSM, LinfDeepFoolAttack, LinfPGD, and PGD, the accuracy drops significantly after epsilon = 0.0008.

a decline in model accuracy. These patterns are consistent across all tested attacks.

Imagenet Dataset: Fig. 9 shows the impact of various adversarial attacks on the RMS values of EM traces under different epsilon values for ImageNet dataset. For each epsilon value, 20 observations were collected to analyze the RMS values under various adversarial attack settings. For instance, in the case of the L2CarliniWagner attack, as shown in Fig. 9c, the RMS values for benign inputs (epsilon = 0.0) lie within a stable range. However, as the epsilon value increases, the RMS values deviate significantly, surpassing the benign threshold.

Similarly, for an attack like FGSM, the RMS values for smaller epsilon values (e.g., 0.0 to 0.01) remain tightly distributed. However, as epsilon increases (e.g., to 0.1 or higher), the RMS values exhibit substantial variations, indicating adversarial interference. For attacks such as LinfPGD, benign inputs at smaller epsilon values (e.g., 0.0 to 0.0008) have consistent RMS values. However, with higher epsilon values, the RMS deviations become prominent, corresponding to a decline in model accuracy.

4.4 Dual-Staged Detection Results

In our RMS based stage-1 detection, however, for certain attacks, such as the LinfDeepFool attack, the deviations in RMS values do not consistently capture adversarial activity, especially for lower epsilon values. In such scenarios, we engage the stage-2 CNN-based detector. In this section, we discuss the results of our dual-staged methodology detection methodology on MNIST, CIFAR10 and ImageNet datasets. The CNN model architecture for detection is implemented in Python as shown below:

```
model = tf.keras.Sequential([
    tf.keras.layers.Input(shape=(train_data.shape[1], train_data.shape[2])),
    tf.keras.layers.Conv1D(16, kernel_size=5, activation='relu'),
    tf.keras.layers.MaxPooling1D(pool_size=2),
    tf.keras.layers.Conv1D(32, kernel_size=5, activation='relu'),
    tf.keras.layers.MaxPooling1D(pool_size=2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(2, activation='softmax')
])
```

We have utilized the Tensorflow framework [7] to implement and train the CNN model for adversarial input detection. The model consists of two 1D convolutional layers with ReLU activation, followed by max-pooling layers to downsample the input. After flattening the output, the model includes a fully connected dense layer with 128 neurons and another dense layer with a softmax activation for classification. The model was compiled using the Adam optimizer with a learning rate of 0.0001 and used categorical cross-entropy as the loss function while monitoring accuracy as the evaluation metric. In this implementation, the preprocessing step involves reducing the dimensionality of the input data by averaging over fixed column intervals. Specifically, we reduce the number of columns in the dataset by computing the mean over every 100 consecutive columns. This approach helps to significantly compress the data while retaining the essential features and variability required for classification. To train the model, we gathered 20 observations for each epsilon value associated with each attack. Epsilons that cause a significant drop in DNN inference accuracy are labeled as adversarial (1), while the rest are labeled as benign (0). The trained model takes in as input the column-reduced traces and performs a binary classification to flag whether the computation is benign or adversarial.

The performance of our dual-staged detection model for different adversarial attacks is summarized in Table 1. The table compares the accuracy of the detection model for MNIST, CIFAR10 and ImageNet datasets across eight adversarial attack methods. In Stage 1, adversarial or benign computations are correctly detected 50% of the time. In Stage 2, the CNN model on an average exhibits a misclassification rate of 17%. As observed, the dual-staged detection methodology achieves high accuracy close to $92-98\%$ for certain attacks, such as FGM, FGSM, L2CarliniWagner, L2PGD, and PGD. For attacks like LinfPGD, LinfDeepFool, and LinfAdditiveUniformNoise the accuracy ranges from $89-90\%$ demonstrating its robustness in identifying adversarial inputs. Despite these variations, the table demonstrates that the dual-staged detection approach provides a reliable means of adversarial detection, achieving high accuracy across different attack scenarios and datasets. It is important to note that computing RMS detection takes approximately 0.46 ms, while inference of the CNN detection model requires about 50 ms. The size of the model is approximately 1.3 MB which makes it suitable for lightweight implementations. We optimized our model using TensorFlow Lite, enabling reduced model size and faster inference while maintaining high accuracy.

Takeaway: A direct comparison with the EMShepherd framework is not possible since their detection approach relies on class-specific features, whereas our detection method is class-agnostic. The authors have shown that their EM-based detector can effectively detect attacks on most classes with over 90% detection accuracy, with some classes demonstrating sub-par detection accuracy (especially with CIFAR10, which performs worse than MNIST with some CIFAR10 classes having a detection accuracy of less than 70%). Our methodology also has similar results (despite being class-agnostic) as shown in Table. 1 with accuracies ranging from $87-98\%$ for different attacks.
If we compare the detection turnaround time, EMShepherd's detection methodology takes 169 ms whereas our RMS-based detection requires only 0.46 ms and CNN-based detection requires an average time of 50 ms. Note that the CNN-based detection is triggered iff RMS detection is unreliable; this implies a majority of adversarial attacks are detected in about 0.46 ms (as compared to 169 ms that EMShepherd takes for *all* attacks).

Table 1: Comparison of Accuracy for Different Attacks using the Dual-Staged Detection Methodology as shown in Fig. 5 for the victim model architectures described in Sec. 4.1.

Attack	Accuracy		
	MNIST	CIFAR-10	ImageNet
FGM	0.931	0.985	0.868
FGSM	0.927	0.938	0.943
L2 Carlini Wagner	0.974	0.934	0.912
L2 PGD	0.961	0.926	0.881
Linf AdditiveUniformNoise	0.877	0.926	0.882
Linf DeepFool	0.935	0.897	0.891
Linf PGD	0.876	0.897	0.905
PGD	0.885	0.925	0.943
Victim Models	Foolbox Zoo [36]	Wong2020FastNet [42]	MobileNet_v2 [14]

5 Conclusion

In this paper, we introduced a dual-staged, lightweight methodology for detecting adversarial attacks in a black-box setting by leveraging hardware-centric characteristics, particularly electromagnetic (EM) emissions. The proposed framework combines a simple yet effective RMS-based detector with a secondary CNN-based classifier for robust detection. Through extensive evaluations on diverse datasets, including MNIST, CIFAR10, and ImageNet, we demonstrated the efficacy of our approach in achieving high detection accuracy ranging from 87% to 98%. Additionally, our method significantly outperforms state-of-the-art solutions like EMShepherd in terms of detection turnaround time, with RMS detection requiring only 0.46 ms and CNN detection averaging at 50 ms, with 1.3 MB memory footprint. These results validate the scalability and practicality of our approach for resource-constrained devices. Our methodology represents a significant step towards real-time, class-agnostic adversarial detection that is deployable in practical edge-device scenarios. Future work will explore extending this approach to more complex hardware platforms, such as Intel chiplets

and Google Coral, and further refining detection mechanisms to enhance robustness against advanced attack vectors. This will ensure secure and reliable deep learning deployments across diverse real-world applications.

References

[1] Mahbubul Alam, Md Imdadul Islam, and MR Amin. 2011. Performance comparison of STFT, WT, LMS and RLS adaptive algorithms in denoising of speech signal. *International Journal of Engineering and Technology* 3, 3 (2011), 235.

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

[3] Anirban Chakraborty, Manaar Alam, Vishal Dey, Anupam Chattopadhyay, and Debdeep Mukhopadhyay. 2021. A survey on adversarial attacks and defences. *CAAI Transactions on Intelligence Technology* 6, 1 (2021), 25–45.

[4] Nikhil Chawla, Arvind Singh, Monodeep Kar, and Saibal Mukhopadhyay. 2019. Application inference using machine learning based side channel analysis. In *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 1–8.

[5] Nikhil Chawla, Arvind Singh, Harshit Kumar, Monodeep Kar, and Saibal Mukhopadhyay. 2020. Securing iot devices using dynamic power management: Machine learning approach. *IEEE Internet of Things Journal* 8, 22 (2020), 16379–16394.

[6] Yen-Chi Chen. 2017. A tutorial on kernel density estimation and recent advances. *Biostatistics & Epidemiology* 1, 1 (2017), 161–187.

[7] TensorFlow Developers. 2022. TensorFlow. *Zenodo* (2022).

[8] Ruyi Ding, Cheng Gongye, Siyue Wang, A Adam Ding, and Yunsi Fei. 2023. Emshepherd: Detecting adversarial samples via side-channel leakage. In *Proceedings of the 2023 ACM Asia Conference on Computer and Communications Security*. 300–313.

[9] Hasan Ferit Eniser, Maria Christakis, and Valentin Wüstholz. 2020. Raid: Randomized adversarial-input detection for neural networks. *arXiv preprint arXiv:2002.02776* (2020).

[10] Reuben Feinman, Ryan R Curtin, Saurabh Shintre, and Andrew B Gardner. 2017. Detecting adversarial samples from artifacts. *arXiv preprint arXiv:1703.00410* (2017).

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

[12] Christian Göttel, Konstantinos Parasyris, Osman Unsal, Pascal Felber, Marcelo Pasin, and Valerio Schiavoni. 2021. Scrooge Attack: Undervolting ARM Processors for Profit: Practical experience report. In *2021 40th International Symposium on Reliable Distributed Systems (SRDS)*. IEEE, 187–197.

[13] Kathrin Grosse, Praveen Manoharan, Nicolas Papernot, Michael Backes, and Patrick McDaniel. 2017. On the (statistical) detection of adversarial examples. *arXiv preprint arXiv:1702.06280* (2017).

[14] Andrew G Howard. 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861* (2017).

[15] Yang Hu, Haoxiang Yuan, Qingyun Sun, Bin Xiao, and Jianxin Li. [n. d.]. Activating More Advantageous Neurons Can Improve Adversarial Transferability. ([n. d.]).

[16] Intel. 2024. Running Average Power Limit (RAPL). https://www.intel.com/content/www/us/en/developer/articles/technical/software-security-guidance/advisory-guidance/running-average-power-limit-energy-reporting.html.

[17] Md Shihabul Islam, Mahmoud Zamani, Chung Hwan Kim, Latifur Khan, and Kevin W Hamlen. 2023. Confidential execution of deep learning inference at the untrusted edge with arm trustzone. In *Proceedings of the Thirteenth ACM Conference on Data and Application Security and Privacy*. 153–164.

[18] Kirthivasan Kannan, B Rajesh Kanna, and Chandrabose Aravindan. 2010. Root mean square filter for noisy images based on hyper graph model. *Image and Vision Computing* 28, 9 (2010), 1329–1338.

[19] Yigitcan Kaya, Bilal Zafar, Sergul Aydore, Nathalie Rauschmayr, and Krishnaram Kenthapadi. 2022. Generating distributional adversarial examples to evade statistical detectors. (2022).

[20] Julian Kunkel and Manuel F Dolz. 2018. Understanding hardware and software metrics with respect to power consumption. *Sustainable Computing: Informatics and Systems* 17 (2018), 43–54.

[21] Etienne Le Sueur and Gernot Heiser. 2010. Dynamic voltage and frequency scaling: The laws of diminishing returns. In *Proceedings of the 2010 international conference on Power aware computing and systems*. 1–8.

[22] Moritz Lipp, Andreas Kogler, David Oswald, Michael Schwarz, Catherine Easdon, Claudio Canella, and Daniel Gruss. 2021. PLATYPUS: Software-based power side-channel attacks on x86. In *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 355–371.

[23] Chen Liu, Abhishek Chakraborty, Nikhil Chawla, and Neer Roggel. 2022. Frequency throttling side-channel attack. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*. 1977–1991.

[24] Zhuang Liu, Ye Lu, Xueshuo Xie, Yaozheng Fang, Zhaolong Jian, and Tao Li. 2021. Trusted-dnn: A trustzone-based adaptive isolation strategy for deep neural networks. In *Proceedings of the ACM Turing Award Celebration Conference-China*. 67–71.

[25] Shiqing Ma and Yingqi Liu. 2019. Nic: Detecting adversarial samples with neural network invariant checking. In *Proceedings of the 26th network and distributed system security symposium (NDSS 2019)*.

[26] Xingjun Ma, Bo Li, Yisen Wang, Sarah M Erfani, Sudanthi Wijewickrema, Grant Schoenebeck, Dawn Song, Michael E Houle, and James Bailey. 2018. Characterizing adversarial subspaces using local intrinsic dimensionality. *arXiv preprint arXiv:1801.02613* (2018).

[27] Aleksander Mądry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2017. Towards deep learning models resistant to adversarial attacks. *stat* 1050, 9 (2017).

[28] Md Mamun, Mahmoud Al-Kadi, and Mohd Marufuzzaman. 2013. Effectiveness of wavelet denoising on electroencephalogram signals. *Journal of applied research and technology* 11, 1 (2013), 156–160.

[29] Dongyu Meng and Hao Chen. 2017. Magnet: a two-pronged defense against adversarial examples. In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*. 135–147.

[30] Aghiles Ait Messaoud, Sonia Ben Mokhtar, Vlad Nitu, and Valerio Schiavoni. 2022. Shielding federated learning systems against inference attacks with ARM TrustZone. In *Proceedings of the 23rd ACM/IFIP International Middleware Conference*. 335–348.

[31] David J Miller, Zhen Xiang, and George Kesidis. 2020. Adversarial learning targeting deep neural network classification: A comprehensive review of defenses against attacks. *Proc. IEEE* 108, 3 (2020), 402–433.

[32] Nimish Mishra, Tridib Lochan Dutta, Shubhi Shukla, Anirban Chakraborty, and Debdeep Mukhopadhyay. 2024. Too Hot to Handle: Novel Thermal Side-Channel in Power Attack-Protected Intel Processors. In *2024 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. IEEE, 378–382.

[33] Shayan Moini, Shanquan Tian, Daniel Holcomb, Jakub Szefer, and Russell Tessier. 2021. Power side-channel attacks on BNN accelerators in remote FPGAs. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 11, 2 (2021), 357–370.

[34] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. 2016. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2574–2582.

[35] FJ Owens and MS Murphy. 1988. A short-time Fourier transform. *Signal Processing* 14, 1 (1988), 3–10.

[36] Jonas Rauber, Wieland Brendel, and Matthias Bethge. 2017. Foolbox: A python toolbox to benchmark the robustness of machine learning models. *arXiv preprint arXiv:1707.04131* (2017).

[37] Alexander J Smola, A Gretton, and K Borgwardt. 2006. Maximum mean discrepancy. In *13th international conference, ICONIP*. 3–6.

[38] Jons-Tobias Wamhoff, Stephan Diestelhorst, Christof Fetzer, Patrick Marlier, Pascal Felber, and Dave Dice. 2014. The {TURBO} diaries: Application-controlled frequency scaling explained. In *2014 USENIX Annual Technical Conference (USENIX ATC 14)*. 193–204.

[39] Yingchen Wang, Riccardo Paccagnella, Elizabeth Tang He, Hovav Shacham, Christopher W Fletcher, and David Kohlbrenner. 2022. Hertzbleed: Turning power {Side-Channel} attacks into remote timing attacks on x86. In *31st USENIX Security Symposium (USENIX Security 22)*. 679–697.

[40] Yingchen Wang, Riccardo Paccagnella, Alan Wandke, Zhao Gang, Grant Garrett-Grossman, Christopher W Fletcher, David Kohlbrenner, and Hovav Shacham. 2023. DVFS frequently leaks secrets: Hertzbleed attacks beyond SIKE, cryptography, and CPU-only data. In *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2306–2320.

[41] Mark Weiser, Brent Welch, Alan Demers, and Scott Shenker. 1996. Scheduling for reduced CPU energy. *Mobile Computing* (1996), 449–471.

[42] Eric Wong, Leslie Rice, and J Zico Kolter. 2020. Fast is better than free: Revisiting adversarial training. *arXiv preprint arXiv:2001.03994* (2020).

[43] W Xu. 2017. Feature squeezing: Detecting adversarial exa mples in deep neural networks. *arXiv preprint arXiv:1704.01155* (2017).

[44] Tianrun Yu, Chi Cheng, Zilong Yang, Yingchen Wang, Yanbin Pan, and Jian Weng. 2024. Hints from Hertz: Dynamic Frequency Scaling Side-Channel Analysis of Number Theoretic Transform in Lattice-Based KEMs. *Cryptology ePrint Archive* (2024).

[45] Peterson Yuhala. 2023. Enhancing IoT Security and Privacy with Trusted Execution Environments and Machine Learning. In *2023 53rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks-Supplemental Volume (DSN-S)*. IEEE, 176–178.

[46] Wei Emma Zhang, Quan Z Sheng, Ahoud Alhazmi, and Chenliang Li. 2020. Adversarial attacks on deep-learning models in natural language processing: A survey. *ACM Transactions on Intelligent Systems and Technology (TIST)* 11, 3 (2020), 1–41.