VERA: VERifiable Microarchitectural Monitoring for Adversarially Robust Edge AI

Soumi Chatterjee, Debadrita Talapatra, Nimish Mishra, Debdeep Mukhopadhyay
Indian Institute of Technology Kharagpur, India

Abstract

As machine learning increasingly moves to edge devices, model owners must trust predictions produced on devices and inputs outside their direct control. This trust is challenged by *adversarial inputs*, where carefully crafted perturbations can induce incorrect predictions. Existing black-box defenses can detect such inputs using microarchitectural signals, but provide no privacy-preserving mechanism for a remote model owner to verify the detection outcome.

In this work, we introduce **VERA**, a framework for verifiable and privacy-preserving adversarial detection at the edge. VERA combines lightweight Hardware Performance Counter (HPC) monitoring with Zero-Knowledge Range Proofs (ZKRPs). Adversarial perturbations can alter internal activation patterns and consequently the microarchitectural behavior of inference, which can be captured through HPC measurements. Rather than revealing these potentially sensitive measurements, VERA allows an edge device to prove that a committed HPC value lies within a calibrated benign range without disclosing the value itself. This avoids the overhead of general-purpose zk-SNARKs and enables lightweight verification on resource-constrained devices.

We formalize VERA as a black-box framework that can combine an HPC-based adversarial detector with an interactive ZKRP, which can also be made non-interactive using the Fiat–Shamir transform. We evaluate VERA against multiple adversarial attacks on MNIST and CIFAR-10. Our results show millisecond-scale verification overhead, with proof-generation costs amortizable across batches of inferences. To the best of our knowledge, VERA is the first framework to provide privacy-preserving cryptographic evidence that an edge inference exhibits microarchitectural behavior within a calibrated benign regime.

Contents

1 Introduction **3**

2 Background **5**

2.1 Adversarial Attacks and Black-Box Detection on Edge AI 5

2.2 Self-Proving Models 6

2.3 Zero-Knowledge Range Proofs 6

2.4 Bulletproofs 7

3 VERA: Verifiable Microarchitectural Monitoring for Edge AI **8**

3.1 Privacy-Preserving Self-Detecting Machine Learning Models 9

3.2 Privacy, Correctness, and Verifiability Guarantees 10

3.3 Interactive Zero-Knowledge Proofs for Privacy-Preserving Detection 12

3.3.1 Privacy-Preserving Proof of Adversarial Detection 13

3.3.2 Private Range Encoding for Zero-Knowledge Verification 14

3.3.3 Privacy-Preserving Range Verification with Bulletproofs 15

4 Microarchitectural Indicator Selection and Calibration **16**

4.1 From Adversarial Perturbations to Microarchitectural Deviations 17

4.2 Requirements for a VERA-Compatible Microarchitectural Indicator 17

4.3 Evaluator Calibration Procedure 18

5 Experimental Results **19**

5.1 Datasets and Models 19

5.2 Hardware Platform 19

5.3 Data Distribution of HPCs 19

5.4 Performance Overheads 22

5.5 Pre-Activation Distribution Analysis 23

6 Conclusion **23**

1 Introduction

Edge Artificial Intelligence (Edge AI) enables real-time machine learning (ML) inference directly on devices such as smart cameras, drones, wearables, and IoT sensors, supporting applications ranging from smart cities and autonomous vehicles to healthcare and industrial automation [12]. By reducing dependence on centralized cloud infrastructure, Edge AI offers lower latency, reduced communication overhead, and improved data privacy [28, 43]. However, deploying ML models on distributed edge devices also increases their exposure to *adversarial attacks*, where carefully crafted input perturbations can induce incorrect predictions while remaining difficult to detect [25, 38, 3, 39].

Unlike centrally managed cloud systems, edge devices often operate in uncontrolled and resource-constrained environments with limited compute, memory, energy, and trusted runtime support [44]. The consequences can be severe in safety-critical applications. For example, a wearable Electrocardiogram (ECG) monitor performing local arrhythmia detection may keep sensitive patient data on-device, preventing a remote healthcare provider from directly validating the input used for inference. Similarly, adversarial manipulation of autonomous-driving models can lead to unsafe decisions and potentially severe consequences [21, 15, 9].

A Case for "Model Owner" Monitoring. There has been recent growing interest within the use-cases of Edge AI wherein the *model owner* takes consequential steps based on model inference outputs. One promising example is Apple's NeuralHash technology [6] that automatically flags users for law enforcement upon inferring signs of abuse. However, evaluators discovered that it was possible to perturb images[1] [7] to confuse NeuralHash into *inaccurately* flagging benign images as signs of abuse, thereby tricking Apple into potentially marking benign users to law enforcement as abuse perpetrators[2]. Another example is YouTube's use of its systems deployed on mobile devices to discover (and potentially block) YouTube accounts engaged in copyright infringements. However, adversarial perturbations [37] to image frames of uploaded videos can evade YouTube's Content ID algorithm to evade detection of copyright infringement[3]. These case studies raise the following question:

With "Model Owners" basing consequential and impactful decisions based on the edge AI inferences, what mechanisms can be developed to increase owner confidence around non-adversarial inferences on the edge, while preserving privacy of the edge device?

A trivial solution to this problem is a two-staged approach: (1) use a detector for adversarial attacks on edge AI [1, 32, 22]; and (2) reveal the raw data used by the detector to model owner for further analysis. However, this trivial solution has serious problems: first, sending data to model owner incurs additional communication overhead on the already constrained bandwidth of edge systems; second, revealing raw execution traces has been known to imply

[1] Akin to the classic definition of an adversarial attack on machine learning [25].

[2] This issue is now fixed in the latest version of NeuralHash with Apple reassuring users of no danger of false flags raised to law enforcement [42].

[3] Such adversarial examples can be found at [18]. These adversarial attacks extend to auditory data as well.

privacy violations [20, 41, 30]; and most importantly, *there is a question on whether the model owner can trust the raw data itself has not been tampered with.*

Our approach is motivated by a connection between adversarial perturbations, internal model behavior, and micro-architectural execution. A prediction-changing perturbation must induce a sufficient change in the model computation to cross a decision boundary, as characterized by Lipschitz-based robustness bounds [27]. Such perturbations can alter intermediate activation patterns, including the states of neurons close to nonlinear activation boundaries. These activation changes can in turn affect micro-architectural behaviour through changes in control flow, sparsity, memory access, and execution timing. Consequently, hardware performance counters (HPCs) provide lightweight mechanism for observing such execution-level deviations.

Given this backdrop, we design and instantiate VERA, a novel framework that enables edge devices to prove to the model owner that inferences were performed on non-adversarial inputs, with *cryptographic* guarantees that (1) the edge device's privacy is preserved and (2) the proof has not been maliciously tampered with. While such guarantees could be realized using zk-SNARKs (as in [4], albeit in a different context), their overhead makes them impractical for resource-constrained edge devices. Instead, VERA leverages microarchitectural behaviour as a lightweight indicator of whether an inference is consistent with a calibrated benign execution regime. Importantly, VERA is not tied to a specific HPC. During an offline calibration phase, the evaluator profiles candidate HPCs on the target model and platform and selects an indicator whose benign measurements form a sufficiently stable range and whose behavior is sensitive to prediction-changing perturbations. The resulting benign range is then encoded into a lightweight, protocol-specific Zero-Knowledge Range Proof (ZKRP) [34]. The edge device can therefore cryptographically prove that an observed HPC measurement lies within the calibrated benign range without revealing the measurement itself.

Contributions: In summary, VERA addresses a key gap in edge ML by combining hardware-based side-channel detection with zero-knowledge proofs to enable privacy-preserving and cryptographically verifiable adversarial input detection.

- We establish a principled connection between prediction-changing adversarial perturbations and microarchitectural observability. Building on Lipschitz-based robustness arguments, we reason that perturbations capable of changing model predictions can alter internal activation patterns, which may propagate to measurable changes in execution behavior. This motivates HPCs as lightweight indicators of benign versus adversarial inference.
- We formulate VERA as an HPC-agnostic framework rather than a detector tied to a specific counter. We identify the properties required of a VERA-compatible microarchitectural indicator and provide an offline calibration procedure for selecting an HPC and deriving its benign operating range on a target model and platform. VERA combines (1) adversarial attack detector, and (2) Zero-Knowledge Range Proof in a black-box. We establish VERA's formal description, as well as detail definitions for correctness and soundness.
- We emphasize the advantages of VERA's black-box composition by instantiating it

through different (interactive) Zero-Knowledge Proofs of Knowledge in a plug-and-play manner. Furthermore, VERA's black-box design allows application of the generic Fiat-Shamir transform to allow non-interactive proofs.

- Finally, we implement VERA and evaluate it using adversarially perturbed inputs (FGSM, PGD and DF) on a CNN trained on CIFAR-10 and MNIST dataset. Our results show reliable detection performance with only around 70 milliseconds of overhead and 608 bytes of proof size, confirming VERA's practicality for real-time edge deployment. Furthermore, VERA's proof generation can be parallelized along with model inferences, effectively amortizing the proof generation cost over a batch of inferences to have a non-existent overhead.

The remainder of this paper is organized as follows: Section 2 provides background on adversarial ML and black-box detection approaches. Section 3 presents the VERA framework and formalizes the Self-Detecting Model. Section 4 motivates the connection between adversarial perturbations and microarchitectural behavior and describes the indicator selection and calibration procedure. Section 5 presents the experimental evaluation of VERA. Finally, Section 6 concludes the paper and outlines future research directions.

2 Background

In this section, we give a brief overview of adversarial attacks, adversarial attack detection on Edge AI, and zero-knowledge proofs.

2.1 Adversarial Attacks and Black-Box Detection on Edge AI

Adversarial attacks perturb inputs to induce incorrect model predictions [47], posing significant risks in safety-critical applications [29]. They may be targeted or untargeted and operate under white-box or black-box threat models [11]. Common attacks include Fast Gradient Sign Method (FGSM) [25], Projected Gradient Descent (PGD) [31], and DeepFool [33].

Existing defenses include input preprocessing [45], adversarial training [25], and external detectors [2]. However, these approaches may incur computational overhead, generalize poorly to unseen attacks, or require access to model internals, limiting their applicability to resource-constrained edge devices [46]. Such threats have also been demonstrated in safety-critical deployments, including unmanned military vehicles [40] and UAV-based infrastructure inspection [35].

Hardware-level monitoring provides an alternative for black-box detection. Adversarial inputs can induce execution-level changes that manifest through microarchitectural events such as cache misses, branch activity, and instruction counts, which can be efficiently observed using Hardware Performance Counters (HPCs). AdvHunter [1] uses such HPC measurements for adversarial detection, but requires repeated inference, increasing latency and

potentially introducing microarchitectural grooming. MIRAGE [14] demonstrates one-shot detection using microarchitectural footprints. Physical side channels have also been explored: DuDAB [13] uses electromagnetic (EM) emissions with a dual-stage detector, while EMShepherd [22] analyzes EM traces but requires additional signal preprocessing.

While these works demonstrate the effectiveness of hardware-level signals for black-box adversarial detection, they primarily provide local detection outcomes. A remote verifier must still trust the reported decision or underlying measurement. They do not provide a privacy-preserving mechanism to cryptographically verify that the observed execution behavior is consistent with a calibrated benign regime. VERA addresses this gap by combining lightweight microarchitectural monitoring with privacy-preserving verification.

2.2 Self-Proving Models

Self-Proving Models [4] introduce an interactive proof-based framework for verifying model outputs on a per-input basis. In this paradigm, the model produces both an output and a cryptographic proof that enables a verifier to check its correctness. The framework provides sound verification, such that incorrect outputs cannot be falsely validated, and establishes correctness guarantees for inputs drawn from the training distribution. The approach has primarily been demonstrated on algorithmic tasks and requires access to model internals during proof generation.

2.3 Zero-Knowledge Range Proofs

Let $(\mathsf{Gen}, \mathsf{Com})$ be a cryptographic commitment scheme, where $p \leftarrow \mathsf{Gen}(1^\lambda)$ denotes the public parameters generated for security parameter λ. A *Zero-Knowledge Range Proof (ZKRP)* allows a prover to prove that a committed value lies within a public interval without revealing the value itself. Formally, a ZKRP is a zero-knowledge proof of knowledge for the relation

$$\mathcal{R}_{\mathsf{range}} = \{((C, u, v), (m, r)) \mid C = \mathsf{Com}(p, m, r) \wedge u \leq m \leq v\},$$

where m is the private committed value, r is the opening randomness, C is the corresponding commitment, and u and v are the public lower and upper bounds, respectively. Thus, (p, C, u, v) is public, whereas (m, r) constitutes the private witness. A ZKRP consists of efficient prover $P(p, C, u, v; m, r)$ and verifier $V(p, C, u, v, \pi)$ algorithms and satisfies the following properties:

- *Completeness.* If (m, r) is a valid witness such that $C = \mathsf{Com}(p, m, r)$ and $u \leq m \leq v$, then a proof π generated by an honest prover $P(p, C, u, v; m, r)$ will always cause the verifier $V(p, C, u, v, \pi)$ to output 1 (accept).
- *Soundness.* If no valid witness (m, r) exists for the statement, no efficient malicious prover can convince the verifier to accept or output 1, except with negligible probability.

- *Zero-Knowledge.* There exists a simulator that, without access to (m, r), can generate a transcript computationally indistinguishable from an honest execution. Hence, the verifier learns nothing about m beyond the fact that the committed value lies in $[u, v]$.

A *non-interactive zero-knowledge range proof (NIZK)* replaces the interaction between the prover and verifier with a single proof π. Such proofs can be realized in the common reference string (CRS) model or, for suitable public-coin protocols, using the Fiat-Shamir transform in the random oracle model. Building on this formulation, in the next section we introduce our Self-Detecting Model, D_θ, where the private committed value m is instantiated by the HPC measurement h. We use C_h to denote the corresponding commitment and π to denote the range proof, enabling privacy-preserving and verifiable adversarial input detection.

2.4 Bulletproofs

Bulletproofs [10] are short zero-knowledge arguments of knowledge that allow a prover to demonstrate that a committed secret value satisfies a given constraint without revealing the value itself. They are particularly useful for *range proofs*, where the prover shows that a committed value belongs to a specified numerical range. An important advantage of Bulletproofs is that they do not require a trusted setup and provide proofs whose size grows logarithmically with the bit length of the range.

Bulletproofs build on Pedersen commitments. Let $\mathbb{G}$ be a cyclic group of prime order q, with independent generators g and $\tilde{g}$. A prover commits to a secret value $m \in \mathbb{Z}_q$ using a random blinding factor $r \in \mathbb{Z}_q$ as

$$C = g^m \tilde{g}^r. \tag{1}$$

The commitment hides m from the verifier while binding the prover to the committed value. The prover can subsequently prove a statement about m without revealing either m or r.

To construct an n-bit range proof, the prover represents m using its binary decomposition

$$\mathbf{a}_L \in \{0,1\}^n, \qquad \langle \mathbf{a}_L, \mathbf{2}^n \rangle = m, \tag{2}$$

where $\mathbf{2}^n = (2^0, \ldots, 2^{n-1})$, $\langle \mathbf{a}_i, \mathbf{b}_i \rangle = \sum_{i=1}^n a_i \cdot b_i$ denotes the inner product between two vectors $\mathbf{a}, \mathbf{b}$ and defines

$$\mathbf{a}_R = \mathbf{a}_L - \mathbf{1}^n. \tag{3}$$

The binary constraints on $\mathbf{a}_L$ are then encoded algebraically. The prover commits to these vectors together with randomly sampled blinding vectors and sends the resulting commitments to the verifier so that the underlying commitment is computationally hiding. The verifier responds with random challenges, which are used by the prover to construct polynomial relations that capture both the binary constraints and the relation to the committed value.

These polynomial relations are reduced to an inner-product relation and verified using the Bulletproof inner-product argument. The inner-product protocol recursively reduces the dimension of the vectors, allowing the proof size to grow logarithmically rather than linearly

with n. Consequently, the verifier can establish that the prover knows an opening of C to a value in the required range without learning the value itself.

The underlying Bulletproof protocol is public-coin: the verifier's challenges are sampled uniformly at random and are independent of the prover's private witness. The interaction can be removed using the Fiat-Shamir heuristic, where the verifier challenges are instead derived from the preceding protocol transcript using a cryptographic hash function, resulting in a non-interactive proof.

3 VERA: Verifiable Microarchitectural Monitoring for Edge AI

In this section, we present VERA, a framework for verifiable microarchitectural monitoring and privacy-preserving adversarial detection at the edge. At the core of VERA, we introduce a *Self-Detecting Model* D_θ that identifies inputs inducing divergent neuron activation patterns, which manifest as deviations in the corresponding micro-architectural measurements from the expected benign range $\mathbb{R}_{\text{benign}}$. The choice of HPC is not fixed by VERA. Rather, it is determined during an offline calibration phase for the target model and hardware platform. The evaluator selects a microarchitectural indicator whose benign behavior is sufficiently stable to define a range and whose measurements exhibit sensitivity to the execution changes induced by adversarial perturbations. Section 4 formalizes these requirements and describes the corresponding calibration procedure. VERA provides cryptographic guarantees for adversarial input detection in a black-box model by employing Zero-Knowledge Range Proofs (ZKRPs) as a black-box proof primitive. Specifically, the Self-Detecting Model proves whether the private HPC measurement h lies within the valid benign range $\mathbb{R}_{\text{benign}}$, without revealing h to the verifier.

At face value, D_θ and the self-proving models introduced in [5] might look similar. However, there are inherent structural differences, notably in the nature of proof and model access mechanisms. In [5], the authors propose a self-proving model that enables inferences to generate formal interactive proofs of their outputs' correctness. This is achieved via interactive proof systems, where the model engages in a multi-round protocol with a verifier. Crucially, their setting assumes white-box access to the model, allowing the verifier to issue queries and receive internal responses. As a result, their approach, while offering strong theoretical guarantees such as worst-case soundness, is computationally intensive and tailored to models whose internal structure is transparent and queryable. In contrast, our proposed self-detecting model operates under a black-box framework, focusing on detecting adversarial inputs by identifying deviations in the distribution of neuron activation patterns. Our framework employs a black-box instantiation of a ZKRP system to cryptographically prove the correctness of detection based on hardware-level side-channel measurements. This design choice enables a lightweight and efficient detection mechanism, providing formal cryptographic guarantees for adversarial input detection without requiring internal access to the model. Unlike [5], we treat both the model and the cryptographic proof system as black boxes, and usage of ZKRPs allow for lightweight proof systems that can be practically deployed on edge devices.

3.1 Privacy-Preserving Self-Detecting Machine Learning Models

Let $\mathcal{X}$ and $\mathcal{Y}$ denote the input and output spaces of the underlying machine learning model, respectively. We denote the model by $M_\theta : \mathcal{X} \to \mathcal{Y}$, where θ denotes the model parameters. For an input $x \in \mathcal{X}$, the model produces an output $y \leftarrow M_\theta(x)$. During the execution of M_θ on x, we observe HPC measurement $h \in \mathbb{R}$ that captures the microarchitectural behavior associated with the inference. We denote this observation by $h \leftarrow \mathsf{HPC}(M_\theta, x)$. Here, HPC$(\cdot)$ denotes an evaluator-selected microarchitectural indicator rather than a specific hardware counter. Depending on the target model and platform, h may correspond to cache misses, branch-related events, instruction counts, or another measurable execution characteristic satisfying the calibration requirements described in Section 4. In our evaluation, we instantiate h using cache misses. Importantly, h is not an output of M_θ; rather, it is a hardware-level measurement observed during the computation of $M_\theta(x)$. We define the benign HPC range as

$$\mathbb{R}_{\text{benign}} = [u, v] \subset \mathbb{R},$$

where u and v denote the lower and upper bounds, respectively, determined from benign model executions. We define the *Self-Detecting Model* D_θ as the composition of the underlying model and a Zero-Knowledge Range Proof system:

$$D_\theta = \{M_\theta, \mathsf{ZKRP}\}.$$

For each inference, M_θ computes the model output y, while the corresponding HPC measurement h is used as the private value in the ZKRP system. The proof system generates a proof π attesting that the private HPC measurement satisfies the benign-range predicate $h \in \mathbb{R}_{\text{benign}}$, without revealing h to the verifier. Let μ denote a probability distribution over the input space $\mathcal{X}$, where $x \sim \mu$ denotes that the input x is sampled according to μ. The Self-Detecting Model enables a verifier V (e.g., the model owner $\mathbf{U}$) to verify whether an evaluator-selected microarchitectural indicator associated with an input $x \sim \mu$ lies within the calibrated benign execution regime. For each input x, the model computes $y \leftarrow M_\theta(x)$ and the corresponding measurement $h \leftarrow \mathsf{HPC}(M_\theta, x)$ is observed. The ZKRP system then generates a proof π for the statement

$$h \in \mathbb{R}_{\text{benign}}.$$

The verifier accepts the inference as exhibiting benign HPC behavior if the proof π is valid; otherwise, it rejects. A pictorial representation of this interaction is shown in Figure 1.

Definition 1 (Self-Detecting Model) *Let μ_{benign} be a benign distribution over the input space $\mathcal{X}$, and let $\mathbb{R}_{\text{benign}} = [u, v]$ denote the valid benign HPC range. For a verifier V, we define the verifiability of D_θ as*

$$\text{ver}_{V,\mu_{benign}}(\theta) := \Pr_{x \sim \mu_{benign}} \left[\langle V, D_\theta \rangle (x) = \mathsf{accept} \right],$$

*where D_θ executes M_θ on x, observes the corresponding HPC measurement h, and invokes the ZKRP system to prove that $h \in \mathbb{R}_{\text{benign}}$. We say that D_θ is β-*Self-Detecting *with respect to V and μ_{benign} if*

$$\text{ver}_{V,\mu_{benign}}(\theta) \geq \beta.$$

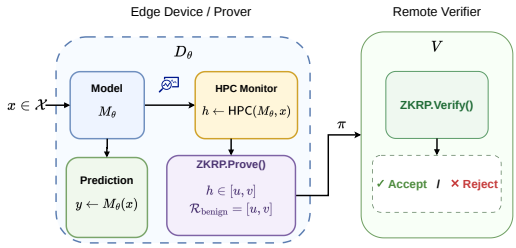


Figure 1: Interactive proof system for privacy-preserving adversarial detection. Given an input $x \in \mathcal{X}$, the Self-Detecting Model $D_\theta = \{M_\theta, \mathsf{ZKRP}\}$ performs inference to obtain $y \leftarrow M_\theta(x)$, while an HPC measurement $h \leftarrow \mathsf{HPC}(M_\theta, x)$ is observed during execution. The ZKRP system generates a proof π attesting that $h \in \mathbb{R}_{\text{benign}}$ without revealing h. The verifier V accepts if the proof π is valid.

where $\beta \in [0, 1]$ denotes the minimum probability with which the verifier accepts a valid benign-range proof for an input drawn from μ_{benign}. Thus, a larger β indicates a higher probability of successful verification by the Self-Detecting Model.

When an interactive ZKRP is instantiated as a black-box proof system, D_θ and V interact to establish the range-membership statement $h \in \mathbb{R}_{\text{benign}}$. Upon completion of the protocol, the verifier outputs either accept or reject. For a fixed input $x \in \mathcal{X}$ and its corresponding HPC measurement $h \in \mathbb{R}$, the verifier's decision may depend on the internal randomness of the underlying proof protocol.

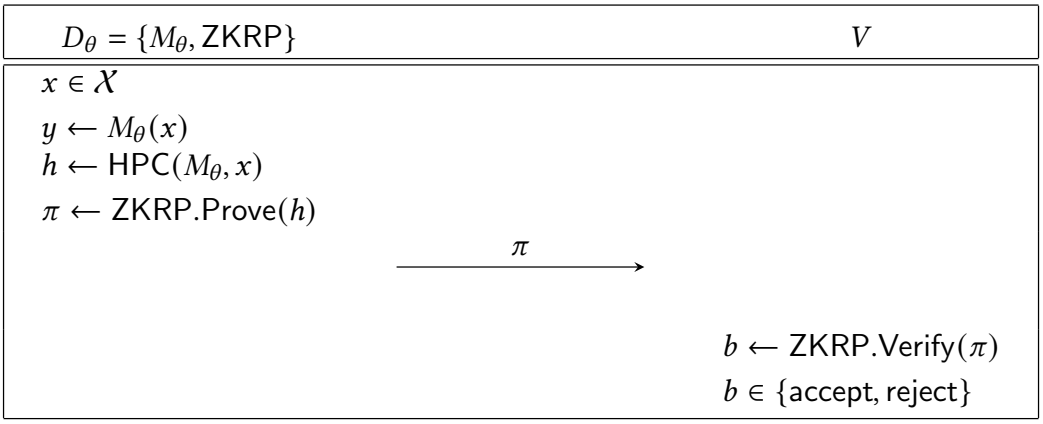


Figure 2: Abstract instantiation of the Self-Detecting Model $D_\theta = \{M_\theta, \mathsf{ZKRP}\}$. Given an input $x \in \mathcal{X}$, the model computes $y \leftarrow M_\theta(x)$, while the HPC measurement $h \leftarrow \mathsf{HPC}(M_\theta, x)$ is observed during inference. The verifier V verifies π without learning h and outputs accept or reject.

3.2 Privacy, Correctness, and Verifiability Guarantees

We first formalize the correctness of the HPC-based detection performed by the Self-Detecting Model. Recall that $\mathcal{X}$ denotes the model input space, μ denotes a probability distribution

over $\mathcal{X}$, and $\mathbb{R}_{\text{benign}} = [u, v]$ denotes the valid HPC range obtained from benign model executions. For an input $x \in \mathcal{X}$, the model computes $y \leftarrow M_\theta(x)$, while the corresponding HPC measurement is observed as $h \leftarrow \mathsf{HPC}(M_\theta, x)$.

Definition 2 (Correctness of Detection) *Let μ_{benign} denote the distribution of benign inputs over $\mathcal{X}$. For any $\alpha \in [0, 1]$, we say that D_θ is α-*correct *with respect to μ_{benign} if*

$$\Pr_{x \sim \mu_{\text{benign}}} \left[h \in \mathbb{R}_{\text{benign}} \,\middle|\, h \leftarrow \mathsf{HPC}(M_\theta, x) \right] \geq \alpha.$$

Here, α denotes the probability that the HPC measurement associated with a benign input lies within the predefined benign range.

Definition 2 separates the statistical validity of the selected microarchitectural indicator from the cryptographic guarantees provided by the proof system. The range $\mathcal{R}_{\text{benign}}$ is meaningful only when the selected HPC exhibits sufficiently concentrated behavior for benign executions and useful separation from adversarial executions. These are empirical properties of the target model-platform pair and are established during calibration, rather than guaranteed by the ZKRP itself.

We next state the cryptographic guarantees required from the proof system used by VERA. For a measurement h, the prover commits to h and proves that the committed value belongs to $\mathbb{R}_{\text{benign}} = [u, v]$, without revealing h to the verifier.

Definition 3 (Completeness, Soundness, and Zero-Knowledge) *Let V be a probabilistic polynomial-time verifier and let D_θ act as the prover in the ZKRP protocol. For the range statement $h \in \mathbb{R}_{\text{benign}}$, the protocol satisfies the following properties:*

- *Completeness. For an honest prover D_θ^* and a valid measurement $h \in \mathbb{R}_{\text{benign}}$,*

  $$\Pr\left[\langle V, D_\theta^* \rangle(h) = \mathsf{accept}\right] = 1,$$

  where the probability is over the randomness of the underlying proof protocol.

- *Soundness. For any efficient malicious prover and a false range-membership statement $h \notin \mathbb{R}_{\text{benign}}$,*

  $$\Pr\left[\langle V, D_\theta \rangle(h) = \mathsf{accept}\right] \leq s,$$

  where $s \in (0, 1)$ denotes the soundness error of the proof system.

- *Zero-Knowledge. The verifier V learns no information about the private HPC measurement h beyond the validity of the range-membership statement. Formally, there exists an efficient simulator that can generate a transcript computationally indistinguishable from an honest protocol execution without access to h.*

The soundness error s bounds the probability that the verifier accepts a false range-membership claim. Thus, for the ZKRP itself, s is the probability that a malicious prover convinces V that a committed measurement belongs to $\mathbb{R}_{\text{benign}}$ when it does not. In VERA, the effectiveness of adversarial detection additionally depends on how well $\mathbb{R}_{\text{benign}}$ separates the HPC

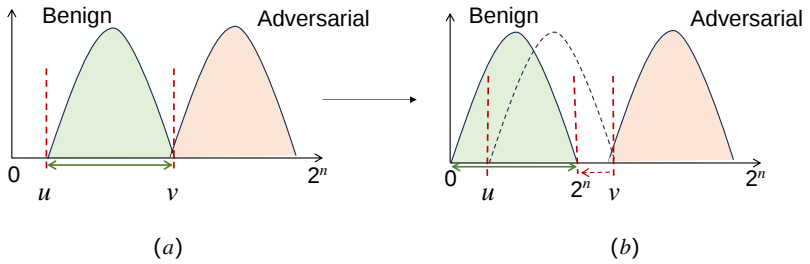


Figure 3: Range normalization in VERA, mapping the model owner's benign HPC range to a range specification supported by a generic ZKRP oracle.

behavior of benign and adversarial inferences. We characterize this separation empirically through the overlap between the kernel density estimations (KDEs) of the corresponding HPC distributions. A smaller overlap indicates better separability between benign and adversarial executions. We analyze this relationship in Section 5.3.

Definition 4 (Verifiability) *Let μ_{benign} denote the benign input distribution over $\mathcal{X}$. The verifiability of D_θ with respect to verifier V and μ_{benign} is*

$$\text{ver}_{V,\mu_{\text{benign}}}(\theta) := \Pr_{x\sim\mu_{\text{benign}}}[\langle V, D_\theta\rangle(x) = \text{accept}] .$$

*We say that D_θ is β-*verifiable *if*

$$\text{ver}_{V,\mu_{\text{benign}}}(\theta) \geq \beta,$$

where $\beta \in [0,1]$ denotes the minimum probability with which a valid benign-range claim produced by the Self-Detecting Model is successfully verified.

Remark. *The correctness of detection and the cryptographic verifiability of VERA capture two distinct guarantees. Correctness characterizes how reliably the observed HPC measurement distinguishes benign behavior, whereas verifiability guarantees that the corresponding range-membership claim can be cryptographically verified without revealing the private measurement h. Consequently, the overall reliability of VERA depends both on the statistical separation of benign and adversarial HPC behavior and on the security guarantees of the underlying ZKRP.*

3.3 Interactive Zero-Knowledge Proofs for Privacy-Preserving Detection

We next instantiate the privacy-preserving detection mechanism of VERA using an interactive Zero-Knowledge Range Proof (ZKRP), enabling the verifier to validate the benign-range membership of the private HPC measurement without learning the measurement itself.

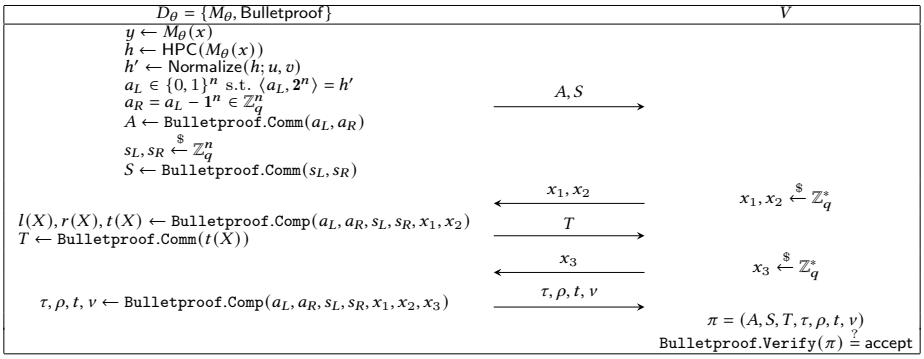


Figure 4: Concrete Bulletproof instantiation of VERA. The verifier accepts if the range proof π for the private HPC measurement h is valid.

3.3.1 Privacy-Preserving Proof of Adversarial Detection

We now describe the abstract protocol underlying the privacy-preserving detection mechanism in VERA. As illustrated in Figure 2, the Self-Detecting Model $D_\theta = \{M_\theta, \mathsf{ZKRP}\}$ consists of the underlying machine learning model M_θ and a ZKRP proof system. Given an input $x \in \mathcal{X}$, the model computes $y \leftarrow M_\theta(x)$, while the corresponding HPC measurement is observed during inference as $h \leftarrow \mathsf{HPC}(M_\theta(x))$. The measurement h remains private and is committed as $C_h = \mathsf{Com}(p, h, r)$, where p is the public parameter and r denotes the commitment randomness. The prover then invokes the ZKRP protocol to prove that the value committed in C_h satisfies $h \in \mathbb{R}_{\text{benign}} = [u, v]$. In an interactive ZKRP, the prover D_θ and verifier V exchange protocol messages according to the underlying proof system. At the end of the interaction, V outputs accept if the range-membership statement is successfully verified and reject otherwise. Importantly, the verifier learns that the committed HPC measurement lies within $\mathbb{R}_{\text{benign}}$ without learning the measurement h itself. The resulting Self-Detecting Model inherits the completeness, soundness, and zero-knowledge guarantees of the underlying ZKRP for the corresponding range-membership statement, as formalized below.

Theorem 1 *If the underlying ZKRP satisfies completeness, soundness with error $s \in (0,1)$, and honest-verifier zero-knowledge for the relation $\mathcal{R}_{\mathsf{range}}$, then the corresponding range-verification protocol instantiated in VERA inherits these guarantees.*

Proof. The claim follows directly from the construction of VERA, which invokes the underlying ZKRP in a black-box manner on the commitment C_h and the public range $\mathbb{R}_{\text{benign}} = [u, v]$. Therefore, completeness, soundness, and honest-verifier zero-knowledge follow from the corresponding properties of the underlying ZKRP. □

Note. If the underlying interactive ZKRP is a suitable public-coin protocol, its interaction can be replaced by a non-interactive proof π using the Fiat–Shamir transform in the random oracle model [8].

3.3.2 Private Range Encoding for Zero-Knowledge Verification

The primary objective of the Self-Detecting Model is to prove the correctness of an HPC measurement after an inference on an adversarial input. This is done by measuring whether the HPC value lies within or outside a client-specified valid range of HPC measurements corresponding to benign inputs. The HPC measurements based on side-channel features, such as cache misses during a model inference, might lie in any arbitrary range depending on the hardware resources used. A generic zero-knowledge range proof verifies the correctness of an HPC measurement by checking whether it lies within a specific range $[0, 2^n - 1]$, for some carefully chosen n. Therefore, it is required to map the client-specified benign range of HPC measurements,

$$\mathbb{R}_{\text{benign}} = [u, v],$$

obtained from running model inference on benign inputs, to a range of the form $[0, 2^n - 1]$, where n is the number of bits required to represent the largest value in the normalized benign range. We call this mapping *range normalization*.

Let us assume that the entire range of HPC measurements corresponding to benign and adversarial inputs is upper bounded by a value B. The range of the benign HPC distribution lies in $\mathbb{R}_{\text{benign}} = [u, v]$, as observed from the hardware measurements (Figure 3a). The range over which a generic ZKRP operates is $[0, 2^n - 1]$, for some carefully chosen n. The normalized range, as shown in Figure 3b, is calculated by shifting the actual benign range. For any $h \in [u, v]$, a normalized measurement $h' \in [0, 2^n - 1]$ is obtained as

$$h' = \frac{h - u}{v - u} \cdot 2^n.$$

Choosing n: Choosing a correct n is critical for ensuring that the transformed value h' lies within the normalized range $[0, 2^n - 1]$ and is correctly verified by a ZKRP oracle[4].

(i) The normalized value must be exact and operate on integers,

$$\frac{2^n}{v - u} \in \mathbb{Z}.$$

(ii) To ensure that the transformed value h' does not overflow, we choose n as the smallest possible integer such that

$$2^n \geq (v - u),$$

$$n \geq \log_2(v - u).$$

Since the proof length of a generic ZKRP scales with the length of n, we choose n as the smallest possible integer such that the above conditions hold. This ensures the minimum bit length of the proof, thereby making VERA lightweight and efficient.

[4]In Figures 2 and 4, we abstract out the range-normalization pre-processing of an HPC measurement h before feeding it into the `ZKRP.Prove` (or `Bulletproof.Prove`) oracle for ease of exposition. However, in the practical implementation, we normalize the actual HPC measurement h to h' such that it can be correctly checked in the pre-computed normalized benign range on which the ZKRP oracle operates.

Since most generic and efficient ZKRPs prove that a value lies in the range $[0, 2^n - 1]$, for some carefully chosen n, and operate over a binary decomposition of the value, the upper bound 2^n must exceed the width of the actual benign HPC range $\mathbb{R}_{\text{benign}} = [u, v]$. Thus, mapping $[u, v]$ to $[0, 2^n - 1]$, where n is the smallest integer such that $2^n \geq (v - u)$, is sufficient and minimal.

3.3.3 Privacy-Preserving Range Verification with Bulletproofs

In our concrete instantiation of Self-Detecting model, we adopt the Bulletproofs protocol [10] as a black-box zero-knowledge range proof oracle, owing to its strong efficiency guarantees and desirable cryptographic properties[5]. ZKRPs are a subclass of zero-knowledge proofs designed to prove that a committed secret value lies within a specified integer interval, without revealing the value itself. A ZKRP enables a prover to convince a verifier that a hidden value, committed using a cryptographic commitment scheme, belongs to a range $\mathbb{R}_{\text{benign}} = [u, v]$.

Christ et al. [16] categorize the underlying techniques used in existing ZKRP constructions [26, 19, 10, 17, 23] into three categories; (i) square decomposition, (ii) binary or n-ary decomposition, and (iii) hash-chain-based approaches. These methods are typically used to prove that a committed value lies in a range of the form $[0, n^k - 1]$, or to establish positivity (as in square decomposition).

Although general-purpose zk-SNARKs (e.g., [26], [24]) can be employed to construct range proofs, they do not leverage the structure of the range-checking relation and are typically less efficient. In contrast, Bulletproofs are a specialized, highly efficient ZKRP construction based on binary decomposition and an inner-product argument. The inner-product protocol compresses proof size, enabling the prover to communicate only $O(\log n)$ group elements for a n-bit range. Bulletproofs offer several additional benefits that make them well-suited for our setting:

(i) *Transparent setup:* No trusted setup is required, making the system easy to deploy.

(ii) *Proof aggregation:* Multiple range proofs can be batched into a single succinct proof.

(iii) *Practical efficiency:* Bulletproofs have short proof sizes $O(\log n)$ and fast verification time $O(n)$, and $O(\log n)$ rounds of interaction (where n is the bit-length of the range), all of which are critical for efficient and lightweight adversarial input detection.

(iv) *Non-interactive:* Since all verifier messages are random elements in the multiplicative group, it can be converted into a non-interactive protocol using the Fiat-Shamir heuristic.

Figure 4 instantiates the abstract Self-Detecting Model of Figure 2 with Bulletproofs [10] as the concrete ZKRP oracle. Given a hardware performance counter (HPC) value h corresponding to model inference, the prover computes a proof π by invoking the `Bulletproof`

[5]We emphasize that our Self-Detecting model can use any generic ZKRP as a black-box and hence one can instantiate it with any ZKRP scheme.

oracle. `Bulletproof.Comm`$(\cdot)$ is a generic representation of the underlying commitment scheme used in Bulletproof, and `Bulletproof.Comp`$(\cdot)$ is used to represent the intermediate computations performed by the prover (see Section 2.4 for more details on Bulletproofs [10]). After running inference to obtain $y \leftarrow M_\theta(x)$ and observing the associated HPC measurement $h \leftarrow \mathrm{HPC}(M_\theta(x))$, the prover D_θ first maps h to the normalized n-bit value $h' \leftarrow \mathsf{Normalize}(h; u, v)$, following the range-encoding procedure described in Section 3.3.2. The prover then represents h' using its binary decomposition $a_L \in \{0,1\}^n$, satisfying $\langle a_L, \mathbf{2}^n \rangle = h'$, and constructs $a_R = a_L - \mathbf{1}^n$. Together, these vectors encode the bit constraints required for proving that the normalized value h' lies within the prescribed range. The prover commits to these vectors as (A, S) to the verifier. The verifier responds with challenge x_1, x_2, which the prover uses to fold the bit constraints and range relations into the polynomials $l(X)$ and $r(X)$ and their inner product $t(X)$. The prover commits to $t(X)$ as T and sends it to the verifier. Upon receiving the final challenge x_3, the prover evaluates the polynomials at x_3 and returns the blinding and evaluation scalars (τ, ρ, t, ν). The resulting transcript $\pi = (A, S, T, \tau, \rho, t, \nu)$ constitutes the range proof, which the verifier checks using `Bulletproof.Verify`(π) and accepts if and only if $h \in \mathbb{R}_{\text{benign}}$, without learning h. Since each verifier message is a uniformly random field element independent of the witness, the interactive protocol can be transformed into a single non-interactive proof using the Fiat-Shamir transform.

The self-detecting model proposed thus, is considered to satisfy the completeness and soundness guarantees as previously described and is zero-knowledge given that Bulletproof is an honest-verifier zero-knowledge range proof with completeness and soundness guarantees. It can be stated as a natural corollary of Theorem 1.

Corollary 1 *If Bulletproof [10] is an honest-verifier zero-knowledge range proof with completeness and soundness guarantees, then VERA is a zero-knowledge self-detecting model with knowledge error* $s \in (0, 1)$.

Proof. Given Bulletproof is a ZKRP with completeness and soundness guarantee and from Definition 2 and 3, with the soundness error bounded by the area overlap between $\mathbb{R}_{\text{benign}}$ and $\mathbb{R}_{\text{adversarial}}$, this holds as a corollary of Theorem 1. □

4 Microarchitectural Indicator Selection and Calibration

VERA relies on microarchitectural measurements as lightweight indicators of whether a model execution is consistent with benign inference. However, VERA is not tied to any specific HPC. Instead, the choice of microarchitectural indicator depends on the target model, hardware platform, and execution environment. In this section, we first provide a principled motivation for why prediction-changing adversarial perturbations can manifest as measurable microarchitectural deviations. We then identify the properties required of an HPC to be suitable for VERA and finally describe an offline calibration procedure through which an evaluator can select an appropriate indicator and establish the corresponding benign range.

4.1 From Adversarial Perturbations to Microarchitectural Deviations

We consider a feedforward neural network $f : \mathbb{R}^d \to \mathbb{R}^K$. For an input x, each neuron produces a pre-activation $z_k^{(i)}(x)$, followed by a nonlinear activation (e.g., ReLU). A neuron is *active* if $z_k^{(i)}(x) > 0$ and *inactive* otherwise. Given a perturbed input $x' = x + \delta$, an *activation flip* occurs when a neuron changes its activation state, i.e.,

$$\text{sign}(z_k^{(i)}(x)) \neq \text{sign}(z_k^{(i)}(x')).$$

Adversarial perturbations induce activation changes. Any perturbation that changes the model's prediction must exceed a minimum magnitude, as established by Lipschitz-based robustness bounds [27]. Such perturbations are expected to alter some neurons across activation boundaries, particularly those with pre-activations close to zero. These neurons form a *susceptible set*, and their transitions result in a non-trivial number of activation flips. Thus, adversarial inputs are expected to induce measurable changes in the network's internal activation pattern.

Activation changes lead to microarchitectural deviations. Although neural network layers access similar memory locations across inputs, execution behavior can differ due to activation-dependent control flow and timing effects. Activation flips can (i) alter branch behavior in nonlinear operations (e.g., ReLU), (ii) introduce sparsity that affects downstream computations, and (iii) cause cumulative micro-timing variations. These changes can manifest through several microarchitectural events, including cache behavior, branch activity, instruction counts, and execution timing. Consequently, activation changes provide a mechanism through which prediction-changing perturbations can induce measurable deviations in HPC measurements.

Microarchitectural observability. Let $\Delta_{\text{HPC}}(x, x') = |h(x') - h(x)|$ denote the deviation in the selected HPC measurement between executions on a benign input x and its perturbed counterpart x'. Activation changes induced by the perturbation may alter execution behavior and consequently produce a measurable Δ_{HPC}. However, VERA does not assume a fixed analytical relationship between the number of activation flips and the resulting HPC deviation. The magnitude of this deviation depends on the model, hardware platform, implementation, and selected HPC. We therefore determine microarchitectural sensitivity empirically through the offline calibration procedure described below.

4.2 Requirements for a VERA-Compatible Microarchitectural Indicator

For an HPC measurement h to serve as a useful microarchitectural indicator in VERA, it should satisfy the following empirical properties.

P1: Benign stability. For benign inputs $x \sim \mu_{\text{benign}}$, the selected indicator h should exhibit sufficiently concentrated behavior such that a meaningful benign operating range

$\mathcal{R}_{\text{benign}} = [u, v]$ can be established. A highly unstable benign distribution would result in a wide range and reduce the ability to distinguish deviations caused by adversarial inputs.

P2: Adversarial sensitivity. Prediction-changing adversarial perturbations should induce a measurable shift in the distribution of h relative to benign execution. In particular, the benign and adversarial distributions should exhibit sufficient separability for range-based detection to be meaningful. The amount of overlap between these distributions therefore characterizes the empirical discriminative capability of the selected HPC.

P3: Deployment stability. The calibrated behavior of h should remain sufficiently stable under the execution conditions for which the verification policy is defined. Since microarchitectural measurements depend on hardware characteristics, model implementation, and execution environment, the benign range is specific to the calibrated model-platform configuration and should not be assumed to transfer unchanged across different devices or implementations.

These requirements concern the *statistical suitability* of the selected microarchitectural indicator and are distinct from the cryptographic correctness of the underlying ZKRP. The ZKRP guarantees that a committed measurement is correctly proven to lie inside or outside the specified range, whereas calibration determines whether membership in that range is a meaningful indicator of benign inference.

4.3 Evaluator Calibration Procedure

Before deployment, VERA performs an offline calibration to select a suitable microarchitectural indicator and establish its benign operating range. For a target model and hardware platform, the evaluator profiles candidate HPCs on representative benign and prediction-changing adversarial inputs and analyzes their stability and statistical separability. An HPC satisfying the requirements in Section 4.2 is selected as the indicator h, and its benign distribution is used to derive $\mathcal{R}_{\text{benign}} = [u, v]$. During deployment, the edge device privately observes h and uses a ZKRP to prove $h \in \mathcal{R}_{\text{benign}}$, without revealing the measurement itself.

An adversarial execution becomes distinguishable when the perturbation induces an HPC measurement that deviates sufficiently from the calibrated benign operating regime. Rather than assuming a fixed analytical relationship between activation changes and HPC deviation, VERA determines this separability empirically during calibration. In our implementation, cache misses are selected due to their stable benign behavior and measurable adversarial shifts. Other HPCs can be used provided they satisfy the stability and separability requirements in Section 4.2.

Takeaway. VERA therefore separates *microarchitectural indicator selection*, which is empirical and platform-specific, from *indicator verification*, which is cryptographic and independent of the particular HPC used. Adversarial perturbations capable of changing model predictions can alter internal activation patterns and subsequently manifest as measurable microarchitectural deviations. An evaluator can exploit this behavior by calibrating an appropriate HPC on the target device, extracting a benign operating range, and using this range as the statement verified by VERA. In this work, we instantiate this procedure using

cache misses, while the framework remains applicable to other HPCs satisfying the above requirements.

5 Experimental Results

In this section we shall be discussing about our results obtained on different adversarial attacks on our target device.

5.1 Datasets and Models

Datasets. We evaluate our approach on MNIST and CIFAR-10. MNIST consists of 70,000 grayscale 28×28 handwritten-digit images across 10 classes, while CIFAR-10 contains 60,000 32×32 color images across 10 object classes. We use Foolbox [36] to generate adversarial samples for both datasets.

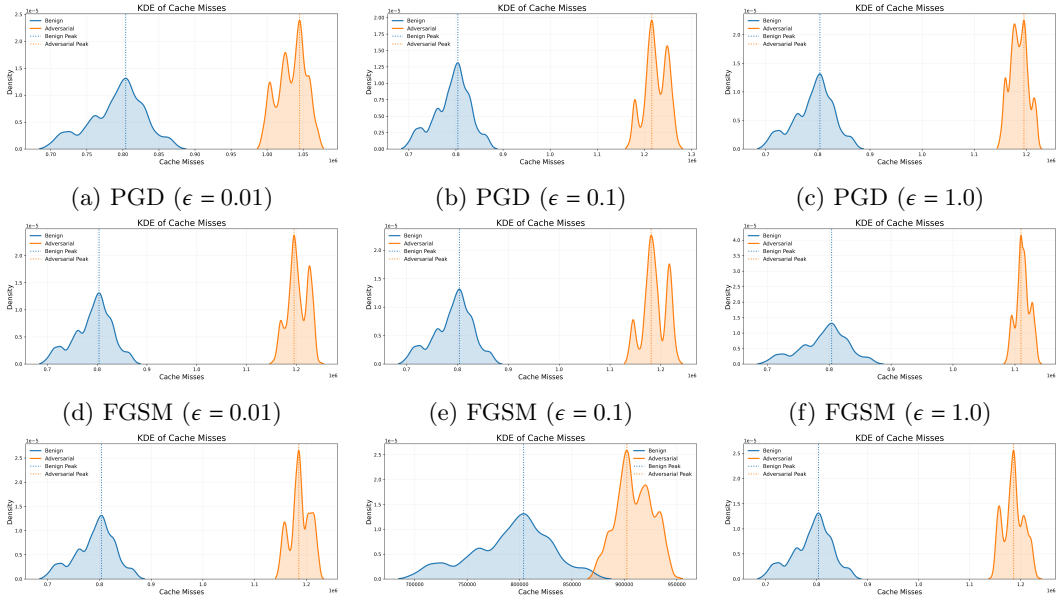
Victim Models. For MNIST, we use a pretrained CNN comprising two convolutional layers (32 and 64 filters), ReLU activations, max-pooling, dropout, and two fully connected layers (128 and 10 units). For CIFAR-10, we use a ResNet-18 model trained on CIFAR-10. The two architectures provide representative victim models of differing depth and complexity for evaluating our proposed approach.

5.2 Hardware Platform

Our experiments were conducted on the Google Coral Dev Board Mini, a compact and energy-efficient edge AI platform tailored for real-time inference in embedded environments. The board is powered by a MediaTek 8167s SoC, which includes a quad-core ARM Cortex-A35 processor and an integrated IMG PowerVR GE8300 GPU. It also features the Google Edge TPU-a dedicated ML accelerator capable of achieving up to 4 trillion operations per second (TOPS) using 8-bit precision, with an energy efficiency of up to 2 TOPS per watt. To monitor microarchitectural events such as cache misses, all neural network inferences were executed directly on the Cortex-A35 cores. This allowed us to leverage tools like `perf` for fine-grained performance monitoring. The board runs on Mendel Linux: a lightweight, Debian-based distribution that offers full system access-enabling precise control over performance measurements and side-channel data collection.

5.3 Data Distribution of HPCs

Following the calibration methodology in Section 4, we instantiate VERA using cache misses as the microarchitectural indicator. Building on the HPC measurement methodology of AdvHunter [1], we focus on cache misses, which exhibit a stable benign distribution and measurable shifts under the considered adversarial perturbations. Unlike *EMShepherd*, which



(a) PGD ($\epsilon = 0.01$) (b) PGD ($\epsilon = 0.1$) (c) PGD ($\epsilon = 1.0$)

(d) FGSM ($\epsilon = 0.01$) (e) FGSM ($\epsilon = 0.1$) (f) FGSM ($\epsilon = 1.0$)

(g) DF ($\epsilon = 0.01$) (h) DF ($\epsilon = 0.1$) (i) DF ($\epsilon = 1.0$)

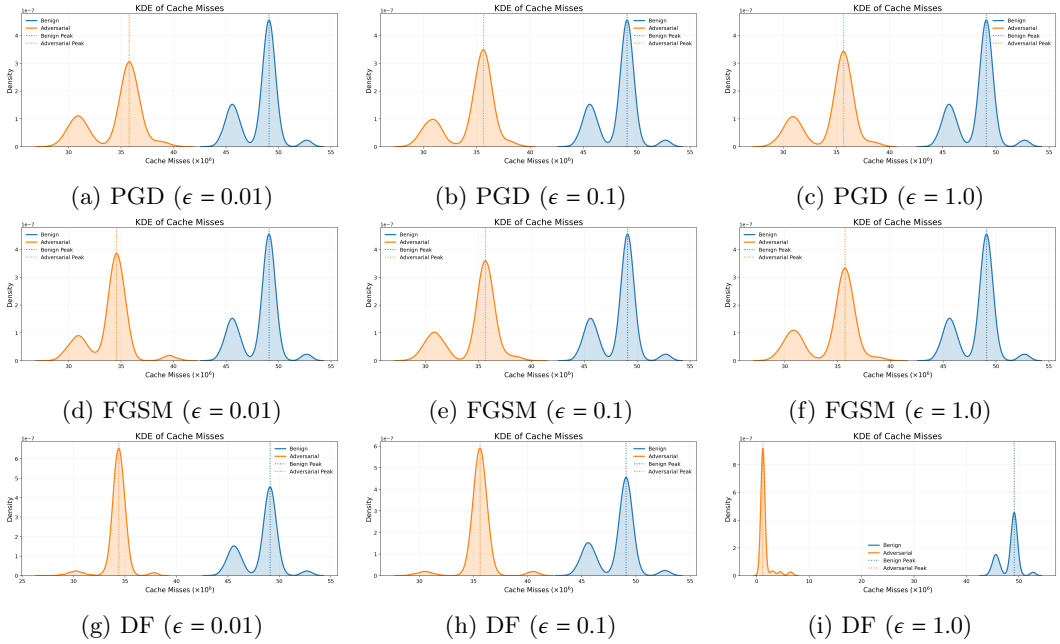
Figure 5: KDE plots of cache misses comparing benign and PGD, FGSM, DF adversarial examples on MNIST for varying ϵ values. **Blue** curves represent benign examples, while **orange** curves represent adversarial examples.

relies on EM side-channel measurements, HPCs can be collected directly on the target device without additional sensing hardware. We modify the AdvHunter implementation[6] to collect cache misses during inference and use kernel density estimation (KDE) to characterize their distribution and construct $\mathcal{R}_{\text{benign}}$.

Figures 5a, 5b, and 5c present the KDE plots of cache-miss counts collected during model inference on the MNIST dataset, under both benign and adversarial conditions. Each figure corresponds to a different perturbation strength (ϵ) used in PGD attacks: (a) $\epsilon = 0.01$, (b) $\epsilon = 0.1$, and (c) $\epsilon = 1.0$. The blue curve denotes the distribution of HPC measurements for benign inputs, while the orange curve corresponds to adversarial inputs. Vertical dashed lines indicate the density peaks for each class. As observed, even at a low perturbation level ($\epsilon = 0.01$), the distributions begin to diverge, though they remain somewhat overlapping. As ϵ increases, the adversarial examples induce increasingly distinct cache-miss patterns, shifting the peak of the distribution further from the benign baseline. At $\epsilon = 1.0$, the separation becomes highly pronounced, with minimal overlap between the two distributions.

As shown in Figure 5d, for FGSM attacks with $\epsilon = 0.01$, the adversarial inputs result in a cache-miss distribution that is noticeably left-shifted compared to benign inputs, with the peak density occurring at a lower value. This trend becomes more prominent at $\epsilon = 1.0$, where the separation between the benign and adversarial distributions is even more pronounced (Figure 5f). The sharp and isolated peak for adversarial examples indicates a

[6]Code: `https://github.com/momalab/AdvHunter.git`



(a) PGD ($\epsilon = 0.01$) (b) PGD ($\epsilon = 0.1$) (c) PGD ($\epsilon = 1.0$)

(d) FGSM ($\epsilon = 0.01$) (e) FGSM ($\epsilon = 0.1$) (f) FGSM ($\epsilon = 1.0$)

(g) DF ($\epsilon = 0.01$) (h) DF ($\epsilon = 0.1$) (i) DF ($\epsilon = 1.0$)

Figure 6: KDE plots of cache misses comparing benign and PGD, FGSM, DF adversarial examples on CIFAR-10 for varying ϵ values. **Blue** curves represent benign examples, while **orange** curves represent adversarial examples.

consistent divergence in hardware behavior induced by the stronger perturbation.

Similarly, Figure 6 illustrates the KDE plots of cache misses for benign and adversarial inputs generated using DeepFool, FGSM, and PGD attacks on the CIFAR-10 dataset. Each row corresponds to a specific attack method, with columns indicating varying perturbation magnitudes ($\epsilon \in 0.01, 0.1, 1.0$). For all attack types, the benign traces exhibit a distinct and consistent peak in the cache-miss distribution, while adversarial examples shift this peak significantly to the right, often forming a separate distribution. Notably, the divergence becomes more pronounced as the attack strength increases, confirming that adversarial perturbations alter the underlying microarchitectural behavior. This clear statistical separation in cache-miss patterns provides a strong signal for differentiating between benign and adversarial inputs using lightweight HPC monitoring, validating our approach's efficacy in black-box edge AI settings.

KDE Overlap and Soundness Error. To quantify the statistical distinguishability between benign and adversarial inferences, we compute the area of overlap between their KDEs over cache-miss distributions. The KDE overlap represents the integral of the pointwise minimum of the two density functions. Intuitively, a higher overlap implies that the adversarial inputs are more likely to mimic the microarchitectural behavior of benign inputs, increasing the likelihood of false proof acceptance. Conversely, a lower overlap indicates stronger separation, leading to reduced soundness error and more reliable detection. This overlap can therefore be interpreted as an empirical upper bound on the verifier's soundness error. In our framework, the soundness error s which quantifies the probability that an adversarial

input with an out-of-range cache-miss measurement is incorrectly accepted by the verifier, is upper bounded by the statistical overlap between the benign and adversarial distributions.

Specifically, we compute this overlap as the area under the pointwise minimum of the KDEs of the two distributions:

$$s \leq \int_{-\infty}^{\infty} \min\left(p_{\text{benign}}(x),\ p_{\text{adv}}(x)\right) dx$$

where $p_{\text{benign}}(x)$ and $p_{\text{adv}}(x)$ denote the estimated probability density functions of HPCs (like cache misses) for benign and adversarial inputs, respectively. A lower overlap implies better separability in the micro-architectural footprint; as seen from the Figure 6 and Figure 5, the overlap is negligible. These distinct shifts in distribution demonstrate that adversarial inputs cause measurable deviations in cache behavior, which can be leveraged as a side-channel signal for attack detection.

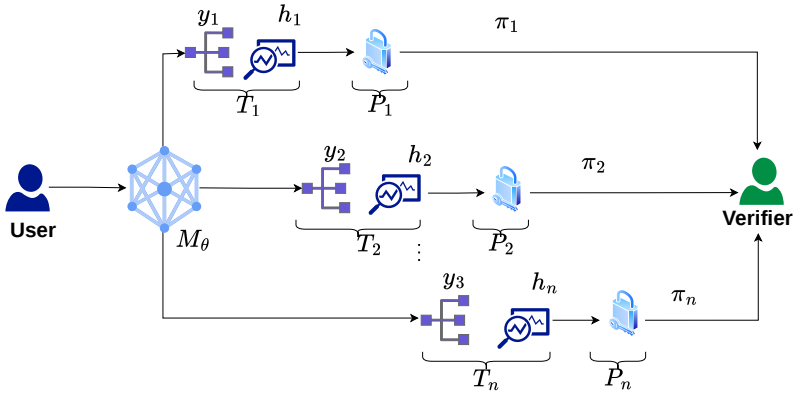


Figure 7: Amortized Overhead for Proof Generation in VERA

5.4 Performance Overheads

Proof Size and Verification Overhead. We measure the cryptographic overhead of the ZKRP component in VERA both in terms of proof size (that directly affects the communication and storage costs) and the computation cost. For PGD, FGSM and DF attacks at perturbation levels $\epsilon = 0.5$ and $\epsilon = 1.0$, the generated ZKRP has a fixed proof size of 608 bytes; this is expected, as the proof size is influenced by the benign distribution only. The average verification time on the verifier's end is 70.69 ms for PGD, 70.32 ms for FGSM and 70.12 ms for DF. These results demonstrate that VERA introduces only minimal verification overhead, making it practical for deployment on latency-sensitive edge AI devices, such as medical wearables.

Amortizing Proof Generation Overhead. Realistically, the edge device shall perform several inferences, which the model owner needs to verify. The design of VERA allows parallelizing the inference and the proof generation in a batch of inferences, effectively amortizing proof generation costs. As shown in Figure 7, let $\{T_1, T_2, \cdots, T_n\}$ represent the time taken for the first, second and n^{th} inferences respectively, and let P_i denote the proof generation time for the zero-knowledge proof of the ith inference. As evident, the proof generation P_i can be subsumed while inference T_{i+1} is being performed, effectively amortizing the proof generation costs over a batch of inferences. This cascading execution reduces the per-inference delay in a streaming setup, and ensures that after the initial overhead, inference and proof generation can proceed in parallel keeping runtime efficient even under proof generation overhead.

5.5 Pre-Activation Distribution Analysis

We analyze how closely neuron pre-activations operate to the ReLU switching boundary. Since a ReLU neuron changes activation state at zero, the absolute pre-activation magnitude $|z_k^{(i)}(x)|$ represents the distance of neuron i in layer k from this boundary. For benign calibration inputs $\mathcal{X}_{\text{cal}}$, we estimate the empirical CDF as

$$F_k(t) = \frac{1}{|\mathcal{X}_{\text{cal}}| n_k} \sum_{x \in \mathcal{X}_{\text{cal}}} \sum_{i=1}^{n_k} \mathbf{1}\left[|z_k^{(i)}(x)| \le t\right], \tag{4}$$

where n_k denotes the number of scalar pre-activation sites monitored in layer, or residual stage, k for each input. Thus, $F_k(t)$ measures the fraction of monitored pre-activations that lie within distance t of the ReLU switching boundary. Figure 8 shows the pre-activation distribution for the MNIST CNN. At $t = 0.1$, approximately 36.43% of `conv1` pre-activations lie near the ReLU boundary, indicating that a non-negligible fraction may be susceptible to activation changes that can propagate to microarchitectural behavior. Additional layer-wise results for MNIST and CIFAR-10 ResNet-18 are provided in Appendix 5.5.

6 Conclusion

In this work, we introduced VERA, a framework for privacy-preserving and verifiable adversarial detection in edge AI. VERA combines lightweight microarchitectural monitoring with Zero-Knowledge Range Proofs (ZKRPs), enabling an edge device to prove that its observed execution behavior lies within a calibrated benign range without revealing the underlying HPC measurement. While we instantiate VERA using cache misses, the framework is HPC-agnostic and allows evaluators to select and calibrate suitable microarchitectural indicators for their target model and platform. Our evaluation on MNIST and CIFAR-10 under FGSM, PGD, and DeepFool demonstrates the feasibility of this approach with low performance and communication overhead.

VERA bridges hardware-assisted adversarial detection with privacy-preserving cryptographic verification, transforming a local detection decision into an externally verifiable claim. Fu-

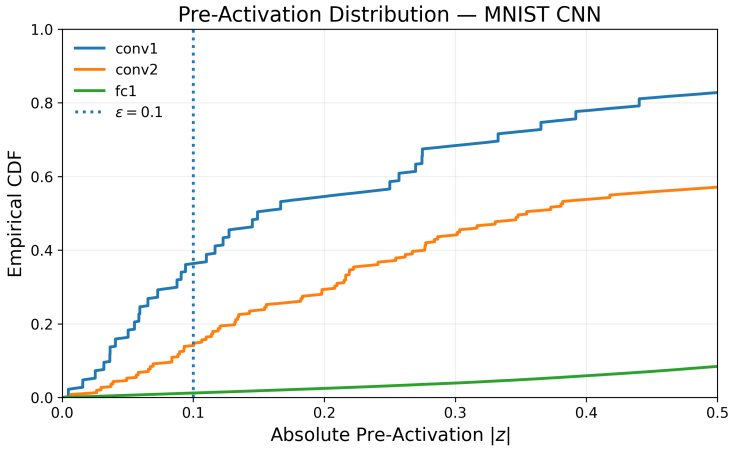


Figure 8: Empirical CDF of pre-activation magnitudes $|z|$ for the MNIST CNN. The dashed line denotes the reference threshold $t = 0.1$.

ture work will investigate adaptive and physically realizable attacks, cross-platform calibration, and richer multi-dimensional microarchitectural indicators and proof mechanisms. More broadly, VERA provides a foundation for trustworthy and verifiable ML inference in resource-constrained and remote edge environments.

References

[1] Alam, M., Maniatakos, M.: Advhunter: Detecting adversarial perturbations in black-box neural networks through hardware performance counters. In: Proceedings of the 61st ACM/IEEE Design Automation Conference. pp. 1–6 (2024)

[2] Aliyu, I., Van Engelenburg, S., Mu'Azu, M.B., Kim, J., Lim, C.G.: Statistical detection of adversarial examples in blockchain-based federated forest in-vehicle network intrusion detection systems. IEEE Access **10**, 109366–109384 (2022)

[3] Alkadi, S., Al-Ahmadi, S., Ismail, M.M.B.: Better safe than never: A survey on adversarial machine learning applications towards iot environment. Applied Sciences **13**(10), 6001 (2023)

[4] Amit, N., Goldwasser, S., Paradise, O., Rothblum, G.: Models that prove their own correctness. arXiv preprint arXiv:2405.15722 (2024)

[5] Amit, N., Goldwasser, S., Paradise, O., Rothblum, G.N.: Models that prove their own correctness. CoRR **abs/2405.15722** (2024). https://doi.org/10.48550/ARXIV.2405.15722, `https://doi.org/10.48550/arXiv.2405.15722`

[6] Apple: Csam detection. `https://www.apple.com/child-safety/pdf/CSAM_Detection_Technica_Summary.pdf` (2021)

[7] AsuharietYgvar: Collisions in neuralhash. https://github.com/AsuharietYgvar/AppleNeuralHash2ONNX/issues/1 (2021)

[8] Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: Denning, D.E., Pyle, R., Ganesan, R., Sandhu, R.S., Ashby, V. (eds.) CCS '93, Proceedings of the 1st ACM Conference on Computer and Communications Security, Fairfax, Virginia, USA, November 3-5, 1993. pp. 62–73. ACM (1993). https://doi.org/10.1145/168588.168596, https://doi.org/10.1145/168588.168596

[9] Boltachev, E.: Potential cyber threats of adversarial attacks on autonomous driving models. Journal of Computer Virology and Hacking Techniques **20**(3), 363–373 (2024)

[10] Bünz, B., Bootle, J., Boneh, D., Poelstra, A., Wuille, P., Maxwell, G.: Bulletproofs: Short proofs for confidential transactions and more. In: 2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings, 21-23 May 2018, San Francisco, California, USA. pp. 315–334. IEEE Computer Society (2018). https://doi.org/10.1109/SP.2018.00020, https://doi.org/10.1109/SP.2018.00020

[11] Carlini, N., Wagner, D.: Towards evaluating the robustness of neural networks. In: 2017 ieee symposium on security and privacy (sp). pp. 39–57. Ieee (2017)

[12] Chang, Z., Liu, S., Xiong, X., Cai, Z., Tu, G.: A survey of recent advances in edge-computing-powered artificial intelligence of things. IEEE Internet of Things Journal **8**(18), 13849–13875 (2021)

[13] Chatterjee, S., Mishra, N., Hazra, A., Mukhopadhyay, D., Chakrabarti, P.P.: Dudab: A dual-staged, lightweight detection methodology for adversarial attacks in a black-box setting. In: 2025 First International Conference on Intelligent Computing and Systems at the Edge (ICEdge). vol. 1, pp. 1–7. IEEE (2025)

[14] Chatterjee, S., Talapatra, D., Mishra, N., Hazra, A., Mukhopadhyay, D.: Mirage: Microarchitectural footprints for detecting adversarial attacks in one-shot inference. In: 2025 IEEE/ACM International Conference On Computer Aided Design (ICCAD). pp. 1–9. IEEE (2025)

[15] Chi, L., Msahli, M., Zhang, Q., Qiu, H., Zhang, T., Memmi, G., Qiu, M.: Adversarial attacks on autonomous driving systems in the physical world: a survey. IEEE Transactions on Intelligent Vehicles (2024)

[16] Christ, M., Baldimtsi, F., Chalkias, K.K., Maram, D., Roy, A., Wang, J.: Sok: Zero-knowledge range proofs. In: Böhme, R., Kiffer, L. (eds.) 6th Conference on Advances in Financial Technologies, AFT 2024, September 23-25, 2024, Vienna, Austria. LIPIcs, vol. 316, pp. 14:1–14:25. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2024). https://doi.org/10.4230/LIPICS.AFT.2024.14, https://doi.org/10.4230/LIPIcs.AFT.2024.14

[17] Chung, H., Han, K., Ju, C., Kim, M., Seo, J.H.: Bulletproofs+: Shorter proofs for a privacy-enhanced distributed ledger. IEEE Access **10**, 42067–42082 (2022). https://doi.org/10.1109/ACCESS.2022.3167806, https://doi.org/10.1109/ACCESS.2022.3167806

[18] CMU: Attacks on copyright systems. https://www.cs.umd.edu/~tomg/projects/copyrightattack/ (2019)

[19] Couteau, G., Klooß, M., Lin, H., Reichle, M.: Efficient range proofs with transparent setup from bounded integer commitments. In: Canteaut, A., Standaert, F. (eds.) Advances in Cryptology - EUROCRYPT 2021 - 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, October 17-21, 2021, Proceedings, Part III. Lecture Notes in Computer Science, vol. 12698, pp. 247–277. Springer (2021). https://doi.org/10.1007/978-3-030-77883-5_9, https://doi.org/10.1007/978-3-030-77883-5_9

[20] Das, S., Werner, J., Antonakakis, M., Polychronakis, M., Monrose, F.: Sok: The challenges, pitfalls, and perils of using hardware performance counters for security. In: 2019 IEEE symposium on security and privacy (SP). pp. 20–38. IEEE (2019)

[21] Deng, Y., Zheng, X., Zhang, T., Chen, C., Lou, G., Kim, M.: An analysis of adversarial attacks and defenses on autonomous driving models. In: 2020 IEEE international conference on pervasive computing and communications (PerCom). pp. 1–10. IEEE (2020)

[22] Ding, R., Gongye, C., Wang, S., Ding, A.A., Fei, Y.: Emshepherd: Detecting adversarial samples via side-channel leakage. In: Proceedings of the 2023 ACM Asia Conference on Computer and Communications Security. pp. 300–313 (2023)

[23] Eagen, L., Kanjalkar, S., Ruffing, T., Nick, J.: Bulletproofs++: Next generation confidential transactions via reciprocal set membership arguments. In: Joye, M., Leander, G. (eds.) Advances in Cryptology - EUROCRYPT 2024 - 43rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zurich, Switzerland, May 26-30, 2024, Proceedings, Part V. Lecture Notes in Computer Science, vol. 14655, pp. 249–279. Springer (2024). https://doi.org/10.1007/978-3-031-58740-5_9, https://doi.org/10.1007/978-3-031-58740-5_9

[24] Gabizon, A., Williamson, Z.J., Ciobotaru, O.: PLONK: permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. IACR Cryptol. ePrint Arch. p. 953 (2019), https://eprint.iacr.org/2019/953

[25] Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572 (2014)

[26] Groth, J.: On the size of pairing-based non-interactive arguments. In: Proceedings, Part II, of the 35th Annual International Conference on Advances in Cryptology — EUROCRYPT 2016 - Volume 9666. p. 305–326. Springer-Verlag, Berlin, Heidelberg (2016)

[27] Hein, M., Andriushchenko, M.: Formal guarantees on the robustness of a classifier against adversarial manipulation. Advances in neural information processing systems **30** (2017)

[28] Li, Y., Ota, K., Dong, M.: Edge ai: On-demand accelerated deep learning inference via edge computing. IEEE Transactions on Industrial Informatics (2018)

[29] Liang, H., He, E., Zhao, Y., Jia, Z., Li, H.: Adversarial attack and defense: A survey. Electronics **11**(8), 1283 (2022)

[30] Lou, X., Chen, K., Xu, G., Qiu, H., Guo, S., Zhang, T.: Protecting confidential virtual machines from hardware performance counter side channels. In: 2024 54th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN). pp. 195–208. IEEE (2024)

[31] Mądry, A., Makelov, A., Schmidt, L., Tsipras, D., Vladu, A.: Towards deep learning models resistant to adversarial attacks. stat **1050**(9) (2017)

[32] Moini, S., Tian, S., Holcomb, D., Szefer, J., Tessier, R.: Power side-channel attacks on bnn accelerators in remote fpgas. IEEE Journal on Emerging and Selected Topics in Circuits and Systems **11**(2), 357–370 (2021)

[33] Moosavi-Dezfooli, S.M., Fawzi, A., Frossard, P.: Deepfool: a simple and accurate method to fool deep neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2574–2582 (2016)

[34] Morais, E., Koens, T., Van Wijk, C., Koren, A.: A survey on zero knowledge range proofs and applications. SN Applied Sciences **1**(8), 946 (2019)

[35] Raja, A., Njilla, L., Yuan, J.: Adversarial attacks and defenses toward ai-assisted uav infrastructure inspection. IEEE Internet of Things Journal **9**(23), 23379–23389 (2022). https://doi.org/10.1109/JIOT.2022.3206276

[36] Rauber, J., Brendel, W., Bethge, M.: Foolbox: A python toolbox to benchmark the robustness of machine learning models. arXiv preprint arXiv:1707.04131 (2017)

[37] Saadatpanah, P., Shafahi, A., Goldstein, T.: Adversarial attacks on copyright detection systems. In: International Conference on Machine Learning. pp. 8307–8315. PMLR (2020)

[38] Shah, P., Govindarajulu, Y., Kulkarni, P., Parmar, M.: Enhancing tinyml security: Study of adversarial attack transferability. arXiv preprint arXiv:2407.11599 (2024)

[39] Sharif, M., Bhagavatula, S., Bauer, L., Reiter, M.K.: Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition. In: Proceedings of the 2016 acm sigsac conference on computer and communications security. pp. 1528–1540 (2016)

[40] Sørensen, S.A.: A Robust and Secure Edge-Based AI System Against Adversarial Attacks. Master's thesis, Oslomet-storbyuniversitetet (2023)

[41] Uhsadel, L., Georges, A., Verbauwhede, I.: Exploiting hardware performance counters. In: 2008 5th workshop on fault diagnosis and tolerance in cryptography. pp. 59–67. IEEE (2008)

[42] Vice (with statements from Apple): Apple defends its anti-child abuse imagery tech after claims of 'hash collisions'. `https://www.vice.com/en/article/apple-defends-its-anti-child-abuse-imagery-tech-after-claims-of-hash-collisions/` (2021)

[43] Wang, S.e.a.: Convergence of edge computing and deep learning: A comprehensive survey. IEEE Communications Surveys & Tutorials (2020)

[44] Wingarz, T., Lauscher, A., Edinger, J., Kaaser, D., Schulte, S., Fischer, M.: Sok: Towards security and safety of edge ai. arXiv preprint arXiv:2410.05349 (2024)

[45] Xu, W., Evans, D., Qi, Y.: Feature squeezing: Detecting adversarial examples in deep neural networks. arXiv preprint arXiv:1704.01155 (2017)

[46] Yi, Z., Qian, Y., Chen, M., Alqahtani, S.A., Hossain, M.S.: Defending edge computing based metaverse ai against adversarial attacks. Ad Hoc Networks **150**, 103263 (2023)

[47] Zhang, C., Benz, P., Lin, C., Karjauv, A., Wu, J., Kweon, I.S.: A survey on universal adversarial attack. arXiv preprint arXiv:2103.01498 (2021)

Additional Pre-Activation Distribution Analysis

This section provides additional layer-wise pre-activation distribution results complementing Section 5.5. For the MNIST CNN, at the reference threshold $t = 0.1$, approximately 36.43% of `conv1`, 14.13% of `conv2`, and 1.22% of `fc1` pre-activations satisfy $|z| \leq 0.1$. Aggregated across the monitored layers, approximately 22.33% of pre-activations lie within this range. These results show that near-boundary behavior is strongest in the earliest convolutional layer and decreases in deeper layers.

For CIFAR-10, we analyze a ResNet-18 model with 94.71% test accuracy and record pre-activation values across the initial convolutional layer and residual stages, namely `conv1`, `layer1`, `layer2`, `layer3`, and `layer4`. Figure 9 shows the corresponding empirical CDFs of $|z|$.

At the reference threshold $t = 0.0015$, approximately 45.99% of `conv1` pre-activations lie within the near-boundary region, whereas the corresponding fractions are 2.43%, 0.57%, 0.68%, and 1.22% for `layer1` through `layer4`, respectively. Aggregated across all monitored stages, approximately 6.84% of pre-activations satisfy $|z| \leq 0.0015$. This again demonstrates strong layer-wise variation: the initial convolutional stage contains a substantially larger fraction of near-boundary pre-activations, while deeper residual stages are more widely separated from the ReLU switching boundary at the same reference threshold.

These observations should be interpreted as a characterization of the pre-activation geometry rather than a direct count of neurons that will necessarily flip under an adversarial perturbation. In particular, an input-space perturbation budget does not directly translate to an equal change in pre-activation space. Nevertheless, the measured distributions identify layers in which a significant fraction of internal values operate close to the ReLU boundary, motivating the subsequent analysis of adversarially induced activation changes and their relationship with microarchitectural behavior.

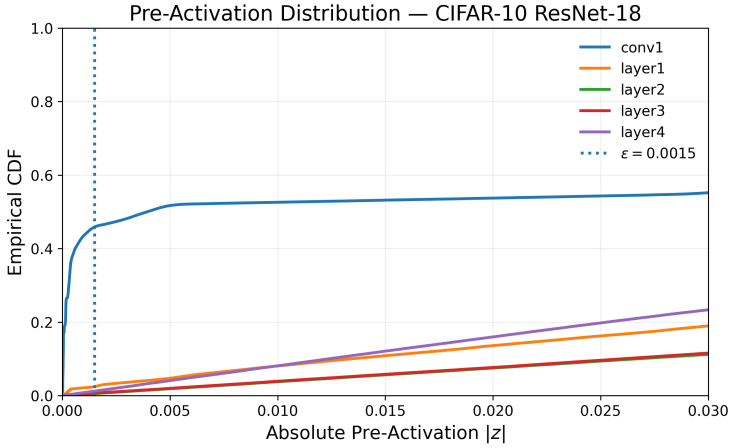


Figure 9: Empirical CDF of pre-activation magnitudes $|z|$ for CIFAR-10 ResNet-18. The dashed line denotes the reference threshold $t = 0.0015$.